WWW.JAMK.FI

CLOUD SERVICE LIFE CYCLE MANAGEMENT IN CONTEXT OF FREENEST SERVICE

Ari Karhunen

Bachelor Thesis
December 2012

Degree Programme in Information Technology
Technology, Communication and Transport

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Author(s)
KARHUNEN, Ari

Type of publication
Bachelor Thesis

Date
06122012

Pages
78

Language
English

Confidential

( ) Until

Permission for web publication
(X)

Title
CLOUD SERVICE LIFE CYCLE MANAGEMENT IN CONTEXT OF FREENEST SERVICE

Degree Programme
Information Technology

Tutor(s)
MANNINEN, Pasi

Assigned by
RINTAMÄKI, Marko – SkyNest project

Abstract

The main objectives of this study were to find the tools that can be used to automate the life cycle of a virtual computer at JunkCloud reference cloud environment that was implemented using OpenStack, and to point out what processes described in IT Infrastructure Library (ITIL) can be used to support the automation process and vice versa.

JunkCloud is SkyNest project's test environment that can be used to test the FreeNest service and which is a playground for the project members to learn from the new technologies. FreeNest is a multiplatform project management tool that is developed by the SkyNest project hosted in the premises of JAMK University of Applied Sciences.

The thesis was carried out by creating a work flow representing the manual steps that were required to start the virtual computer and FreeNest service to the JunkCloud environment. When the work flow was created, the ITIL processes were also referred. Using the activities described in the workflow as a basis, the tools that can be used to create the activities in automated manner were discovered and represented.

This study can be used to support the actual work to create the automated solution, although this study does not provide all or the best solutions because of the broad coverage of the subject, it points to the right direction where to find more information.

Keywords
ITIL, OpenStack, FreeNest, Cloud, IAAS, SkyNest, Python

Miscellaneous

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) KARHUNEN, Ari	Julkaisun laji Opinnäytetyö	Päivämäärä 06122012
	Sivumäärä 78	Julkaisun kieli Englanti
	Luottamuksellisuus ( ) Saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
PILVIPALVELUN ELÄMÄNKAAREN HALLINTA FREENEST-PALVELUN YHTEYDESSÄ

Koulutusohjelma
Tietoverkkotekniikka

Työn ohjaaja(t)
MANNINEN, Pasi

Toimeksiantaja(t)
RINTAMÄKI, Marko - SkyNest project

Tiivistelmä

Opinnäytetyön tarkoituksena oli etsiä tarvittavat työkalut, joita voidaan käyttää virtuaalitietokoneen elämänkaaren automatisoinnissa OpenStack-pohjaisessa JunkCloud-testausympäristössä. Lisäksi työssä oli tehtävänä osoittaa mitä IT Infrastructure Library (ITIL) prosesseja voidaan käyttää automatisoinnin tukena ja mitkä ITIL-prosessit voivat saada tukea automatisoinnilta.

JunkCloud on SkyNest-projektin testiympäristö, jota voidaan käyttää FreeNest-palvelun testaamiseen ja on projektiryhmän pilvipohjainen testiympäristö uusien teknologioiden ja ratkaisujen testaamiseen. FreeNest on SkyNest-projektin kehittämä monialustainen projektinhallintatyökalu ja itse projekti toimii Jyväskylän ammattikorkeakoulun sisällä.

Työ aloitettiin tekemällä työvuokaavio joka esittää tarvittavat käsin tehtävät toimet virtuaalikoneen ja FreeNest-palvelun käynnistämiseksi JunkCloud-ympäristöön. Työvuokaavioiden vaiheita käytettiin myös osoittamaan mitä ITIL-prosesseja voidaan käyttää hyödyksi kyseisissä kohdissa. Työvuokaavioiden pohjalta ehdotettiin työkaluja, joiden avulla kyseinen aktiviteetti voidaan tehdä automatisoidusti.

Opinnäytetyötä voidaan käyttää automatisointiratkaisun rakentamisen tukena, vaikkakaan työ ei tarjoa kaikkia tai parhaita ratkaisuja aiheen laajuuden takia, opinnäytetyö ohjaa oikeaan suuntaan tiedon löytämiseksi.

Avainsanat (asiasanat)
ITIL, OpenStack, FreeNest, Cloud, IAAS, SkyNest, Python

Muut tiedot

CONTENTS

FIGURES

ABBREVATIONS AND ACRONYMS

API	Application Programming Interface is a definition for a term how different applications communicate to each other.
BLOB	Binary Large Object is a large file that requires special handling when placing in to database backend due to its large size. BLOB-storage is a database-backend that is capable of handling these large objects.
Cloud	Cloud is a term to abstract a cluster of computers or a system and the exact definition differs based on subject and point of view. It is something that just works.
Cloud-init	Ubuntu package for managing early initialization of instance at cloud environment.
EC2	Amazon Elastic Compute Cloud
Euca2ools	Is a toolset for managing instances at ec2 type cloud. Instances at OpenStack environment can be managed with these tools.
Fabric	A library for python that provides tools to execute local and remote shell commands.
GlusterFS	Gluster File System is an open source, network file system that is capable to scale up to petabytes in storage.
GUI	Graphical User Interface
IAAS	Infrastructure as a service is a cloud model that provides either physical or virtual machines as a service along with network environment and possibility to manage the environment to a certain degree.
Instance	Term for a virtualized computer in a cloud defined by Amazon.

ISO/IEC 20000 International Standard for IT Service Management.

ITIL	Information Technology Infrastructure Library, is a collection of practices, or a framework, that can be used to support IT services for the business.
JSON	Java Script Object Notation, simple way to transfer human readable data.
KVM	Kernel based Virtual Machine is support build in Linux kernel that enables virtualization to happen in hardware rather than in software.
OpenStack	This is open source project that aims to deliver tools to create a private or public cloud with common hardware. More detailed information in this thesis.
PAAS	Platform as a service is a cloud model that provides computing platform along with operating system as a service
Ping	A nickname for ICMP-echo request defined in RFC 792. Ping is a tool found in different systems to make a ping request to target a host machine in network that in term replies the ping request with pong reply, more accurate term would be ICMP-reply. This tool can be used to verify end-to-end network connectivity to a certain degree and can be used as a debug tool.
Project	Alternative term is “tenant”. At OpenStack environment this means isolated resource container containing VLAN, volumes, instances, images, keys and user. Project can also be described to be a sandbox, and within one sandbox there is interconnectivity, but not to other sandboxes.
Python	A high level dynamic programming language.
Rest/ful	Representational State Transfer, rest, is architecture model using HTTP protocol to gain additional functionalities from it. Within the rest architecture there is a way of reusing the HTTP protocols vocabulary (GET POST PUT DELETE) and to add more information within these simple messages. Restful service is a service that uses HTTP protocol and principles of rest.

Volume	Term for virtual hard drive located in network.
VT-support	Is a term for Intel CPU to have a support for Virtualization Technology that is required to run virtual machines in hardware.
WSGI	Web Server Gateway Interface, interface between web server, web application or framework.
XML	Extensive Markup Language, document format that is in human and machine readable format.

1 BACKGROUND

1.1 Concepts of cloud computing

Cloud computing is a term that can be nowadays heard almost everywhere where IT comes into play and new types of services are made that are distributed from cloud for the end users. The management of the cloud infrastructure is one of the key concepts in this thesis and the thesis begins by describing the key cloud service models and then moves to describing the OpenStack, one of the possible tools to deploy cloud infrastructure. OpenStack is the solution decided to be used in the reference environment and there is a need to simplify the launching and managing of the virtual computers in this environment to free up human resources to other relevant tasks. The goal is to describe steps to manage the virtual computer in the infrastructure, and then based on these steps find out the tools with which the management can be simplified using scripts. In this thesis ITIL is also taken into notice at the point when the workflows are made for the scripts to point out what processes and activities described in ITIL can be used to help the automating process.

1.2 Assigner of thesis and SkyNest

The thesis was assigned by the SkyNest project that works within JAMK University of Applied Sciences; the project is founded by Tekes and is also one of the side projects of ICT-SHOCK's cloud software program. (JAMK ICT, n.d). Within the SkyNest project, one of the goals is to develop the FreeNest portable project platform that has all the necessary tools to house a project from the very beginning to the end of the project's life. The FreeNest itself is built by gluing together open source tools required to handle different tasks in project management and collaboration. The glue for these tools is the FreeNest web based GUI, that can be used to manage and use all the tools found in FreeNest. Another goal of the SkyNest project is to research how the

so called cloud works and find out the tools required to build and manage an own private or public cloud. This cloud research is derived from the idea that the FreeNest Portable Project Platform would be something that could be later delivered as PaaS-solution from the cloud, Platform as a service. To accomplish this need, the students within the project work for their internship with given tasks to find out and learn how to use the tools to build a cloud cluster and then build it in practice for testing and learning purposes. One of the results from this task is the emerging of the reference testing environment called “JunkCloud”, which is described later.

The cloud software program is a research co-operation initiative by TIVIT. The goal of the program is to raise Finland’s position in software development in the global markets, and the cloud software program especially aims at creating new models of business, lean software enterprise models and cloud software infrastructure (TIVIT - Cloud Software Finland, n.d). The TIVIT is a Finnish strategic center for science, technology and innovation in the field of ICT, and is owned by companies and public corporations.

1.3 Junk Cloud as a reference environment

The “JunkCloud” is an infrastructure made using cheap computers that have been salvaged wherever it was possible to find them. The only requirements for those machines were support for virtualization in the CPU and that they were at least somehow functional. The infrastructure is administrated by students doing their internship or thesis for the SkyNest project. The JunkCloud environment is more like a learning environment rather than a production environment and thus it gives the students freedom to test and use software that might even still be under development or not ready for live environment. For the students this is a great learning environment as there is no real stress if the environment breaks because there will be no monetary losses and the environment can be brought back up from scratch in less than a day.

The operating system used in JunkCloud is Ubuntu server and the software used to create the cloudlike abilities to the infrastructure is OpenStack. OpenStack is described later in this thesis in more detail. Illustrative figure (see figure 1) of JunkCloud represents the cluster where the machines are interconnected through a single 48-port 100Mbps switch and one server acts as a gateway to the internet. All together there are around twenty machines that are all participating in some role in the infrastructure, the machines without VT-support are participating to provide the media backend to the infrastructure and the rest of the machines that have the VT-support are participating in an active role for providing the cloud computing abilities to the JunkCloud environment. Although the performance of the environment is relatively poor in every aspect, it suits well as a testing and learning environment.

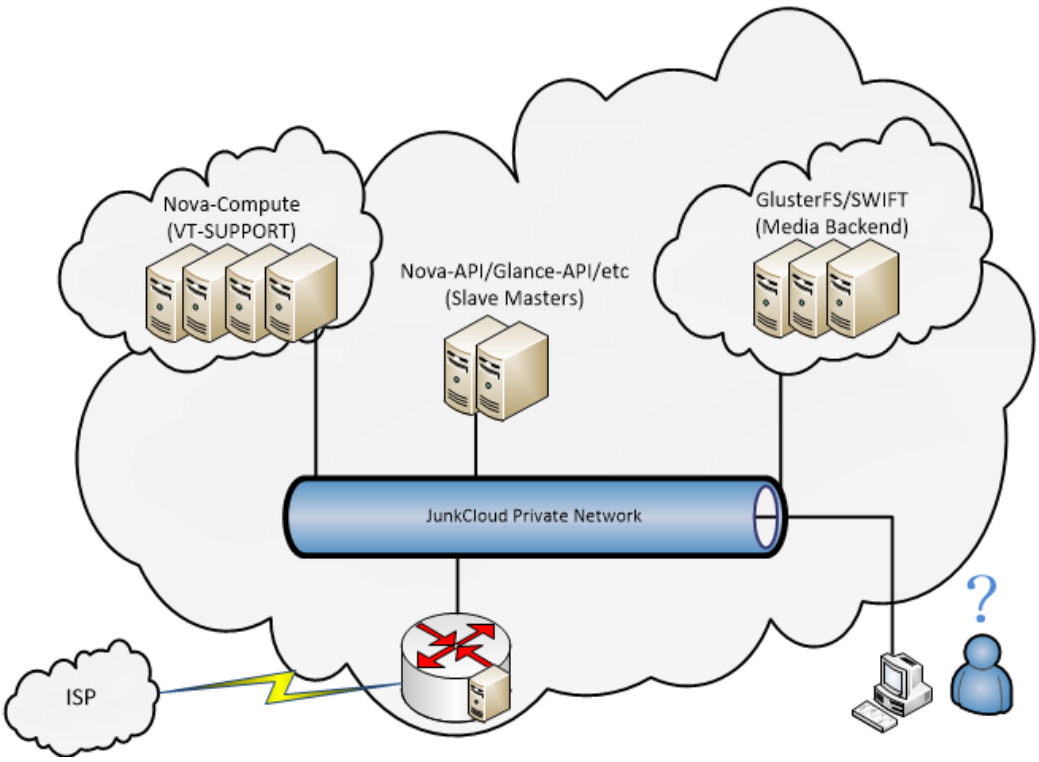

FIGURE 1. Illustrative figure of JunkCloud

1.4 Research objective

The objective of this thesis was to find the tools with which the launching and terminating virtual computers at the JunkCloud environment can be automatized. In addition to launching and terminating, the tools for the running phase of a virtual computer need to be discovered; within the running phase, the backup solution and error recovery are on focus. The first objective of this thesis is to plan and create work flows for the different stages of instance's life that are then used as a reference to find out the tools to automatize the instance's life at JunkCloud-infrastructure. To reach this objective, the first step is the ground work to describe how the infrastructure can be manually managed and to describe the actions and/or commands used, especially the parts of commands that have different variables that affect the functionality of instance in some relevant way. At the same time as the planning goes on, it should be kept in mind that the work flow should also be suited as tool for ITIL's process, for example the event management that can be found in ITIL. After the work flows are created, the second objective is to find out the tools that can be used to script the activities in described work flows. Within this work, the objective is not to plan or create a fully automated infrastructure, but to just represent the tools that are required to launch and terminate a virtual computer and are relevant when the virtual computer is running in the environment.

2 CLOUD SERVICES

2.1 Something as a Service

When running into a term that describes something being delivered as a service, the "as a Service" usually informs right away that this "something" is delivered from the cloud. The term "cloud" is reference to a highly automated

infrastructure that has been used to deliver the service and all of this is abstracted under a single term or a figure to hide the complexity that lies beneath the hood, and some of the different services can be seen in figure 2. where the different services are within the cloud figure and the clients using the cloud services are represented outside the cloud figure. Usually the cloud services that are distributed from the cloud can be divided into different service models or types of service clouds depending on what kind of service is provided for the end user. The most common types of cloud service models today are IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service); these are described in more detail later. (Wikipedia - Cloud Computing, n.d).

Along these three most common service models, there are a lot of additional models and new ones are made up as time goes by. Probably one the most commonly used services is the Dropbox that provides the STaaS (Storage as a service) for the end users.

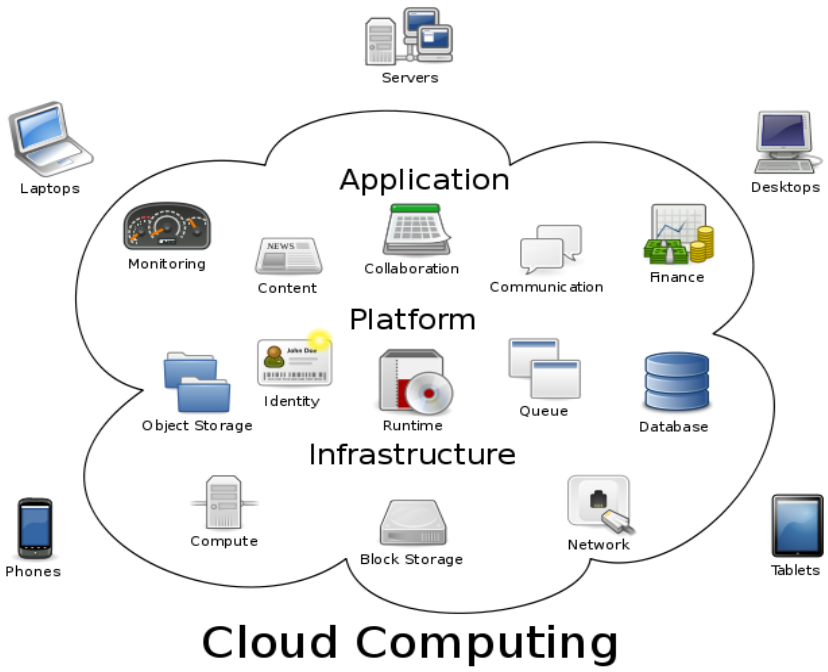

FIGURE 2. Services provided from cloud (Wikipedia – Cloud computing)

2.2 Infrastructure as a Service

Infrastructure as a service (IaaS) is one model of cloud computing where the cloud usually provides either virtual or physical computers for the end user on demand basis. Along with these virtualized platforms, the cloud provides other controllable resources for the user, such as networking to interconnect the computers, storage resources and other relevant resources for the virtual computers to deliver the designated services from the cloud. The benefits of the IaaS-model are: releases resources from managing the IT infrastructure, dynamic scaling, smaller investments on required hardware and utility of used services. The best known IaaS provider is the Amazon Web Services that provides the computing services from their EC2 type cloud. (Wikipedia - Cloud Computing, n.d), (theResearchpedia, n.d)

2.3 Platform as a Service

Platform as a Service (PaaS) is a cloud service model where a service provider delivers just the computing platform for the user and the management of the underlying infrastructure is left for the service provider. The platform can then be used to develop, build, test and deliver the wanted services without needing to handle the infrastructure. Some benefits of PaaS model are for example reduced costs and scaling of resources. One of the best known PaaS providers is the Google with the Google App Engine, with which web services can be developed and hosted. (Wikipedia - Cloud Computing, n.d)

2.4 Software as a Service

Software as a Service (SaaS) is a model where software is delivered on demand from the cloud without the user to be able to manage the underlying infrastructure or the platform from where the service is provided. For the end user, this means that there is no need to handle issues regarding

infrastructure or server management and configuration; the software can be used from anywhere and anytime via web browser or another special tools. The best known SaaS provider is Google with the Google Apps. (Wikipedia - Cloud Computing, n.d)

3 INFRASTRUCTURE

3.1 Cloud Infrastructure

The cloud infrastructure is the foundation of cloud computing and supports all the services that are delivered from the cloud. The cloud infrastructure can be rented as a service from the service provider, but there are solutions to build your own cloud infrastructure. To name a few commercial tools that can be used to build the cloud infrastructure are the wmware's vSphere solution, Microsoft's Hyper-V, Citrix's CloudPlatform and Oracle's cloud solutions. There are also non-commercial solutions for building your own cloud infrastructure, and to name a few there are the OpenNebula's project, Eucaluptys and OpenStack project. From the open source solutions, the OpenStack is the solution that is already in use in the JunkCloud environment and it is described later in better terms.

3.2 OpenStack

3.2.1 What is OpenStack

OpenStack is an open source collaboration divided into three main projects to create a software family that has the capabilities of creating a public and/or private cloud infrastructure using only common hardware. These projects within the OpenStack are called computing, networking and storage, and they are described in the next chapters of this thesis. The OpenStack platform is

intended to be massively scalable, easy to operate and rich in features. There are over 180 companies collaborating in the OpenStack project and it has been founded by NASA and Rackspace Hosting. (OpenStack project, n.d)

3.2.2 OpenStack Compute

Openstack Compute, codename nova, is software to manage and allocate virtual computers in networked server environment. Nova handles all the required tasks needed to start virtual computers to the environment and provides the tools to control the cloud environment.

The nova itself is divided into smaller components that can be then distributed and duplicated to different machines creating redundancy. These smaller components are orchestrated by a component called nova-api that makes the calls to the other components by using queue-server called RabbitMQ. Therefore, as long as the nova-api knows where all the smaller components are located, it can use the services those smaller components provide. The other main components in nova are nova-scheduler, nova-network, nova-compute and nova-volume. The role of nova-scheduler is to provide the nova-api information where there are enough free resources to start the instance. The next component is nova-network whose task it is to create rules relating to networking and allocate private and public addresses to the upcoming instance. Nova-volume's role in this system is to provide virtual hard drives for the instance that can be mounted to the upcoming instance. After consulting these components, nova-api then informs the nova-compute to start the task to run the virtual computer using the credentials and configurations that it defines for the nova-compute to use. After receiving the information, nova-compute then gets the image required to run the instance using OpenStack's imaging service and boots up the virtual computer (Pepple, 2011) (OpenStack Project, n.d). All the components that the nova hides beneath the hood can be seen in figure 3.

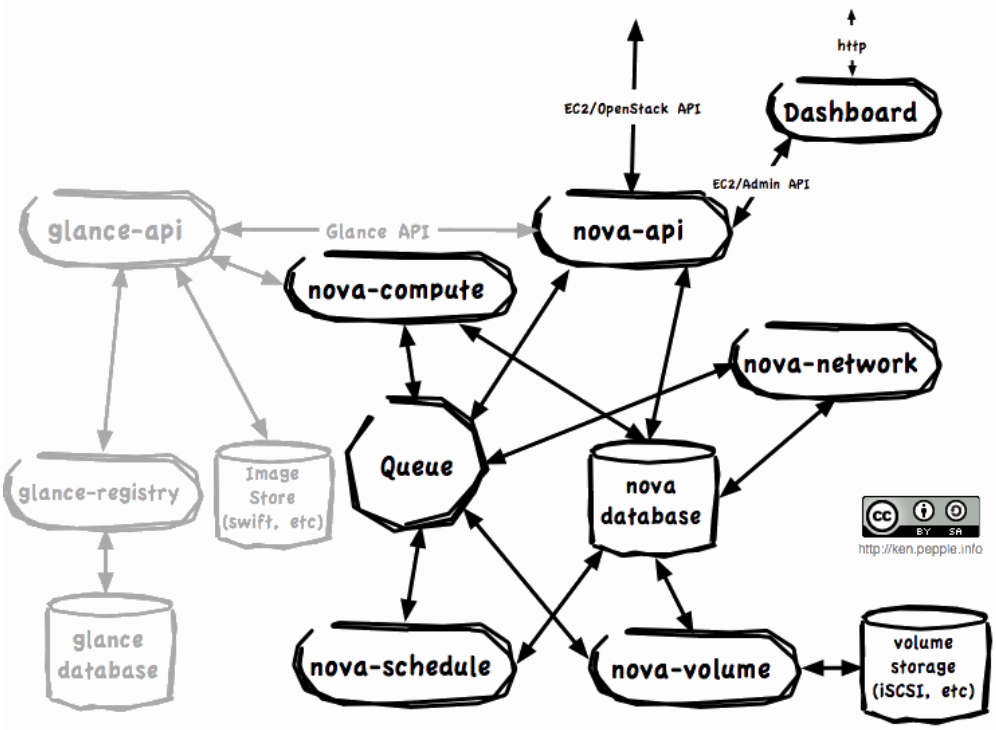

FIGURE 3. OpenStack nova architecture (Pepple, 2011)

3.2.3 OpenStack Image Service

OpenStack Image service, codename glance, is software to work as a middleware between nova-compute and database backend where the images are located and delivers the data for the nova-compute. The glance is divided into two main components called glance-api and glance-registry. Glance-registry's function is to store information where the image data can be found, and the function of glance-api is to relay that information between nova-compute and glance-registry (OpenStack Project, n.d). The relationships between the described services can be seen in the previously represented figure 3.

3.2.4 OpenStack Object Storage

OpenStack object storage, codename swift, is software to create a database backend for storing data. The data stored into swift is considered to be long term, meaning it is not a tool for distributing data backend for real time data

that could be changing constantly. The meaning of “long term” in this case is that the data located in the backend is static data that can either be retrieved or updated if necessary. Swift itself is responsible for the integrity of data located in the database in a way there should never be a situation where the data is lost from the backend. This integrity is accomplished within swift by data replication, eliminating the centralized control of the database backend, and automatic error resolving in a situation where one or more of the nodes in swift should fail.

The swift itself consists of different main components: swift-proxy, object-server, container-server and account-server. The proxy’s role is to relay information between other parts working in swift and act as a relay to public-API. The object-server functions as BLOB-data storage for the swift responsible for saving, deleting and modifying data in local system. Container-server is responsible for handling the listing of objects in different container, it does not actually know where those items are physically, but it is well aware of what objects do belong on what container and then replicates this information to other container servers. Account-server’s function is to be aware and keep a list of containers that are at use. (OpenStack Project, n.d)

3.2.5 Other OpenStack Services

OpenStack identity service, codename keystone, is works as a common authentication channel for all the different components in OpenStack infrastructure. Another commonly used OpenStack service is the Dashboard that provides the web interface to launch and manage the virtual machines in the infrastructure.

4 ITIL

4.1 What is ITIL

Information Technology Infrastructure Library, ITIL, is a collection of practices that provide a framework of good practices and guidance to produce and manage IT-services. The main focus on ITIL practices is the management of IT-services through processes. This framework is made so that it covers the whole lifecycle of the IT-process and within the core of this lifecycle functions the service strategy and it guides the designing, transition and operating the service management lifecycle and all of this is surrounded by continuous service improvement as can be seen in figure 4.The ITIL models describe the goals, general activities, inputs and outputs of the processes which can be used within IT organizations. “ITIL provides a proven method for planning common processes, roles and activities with appropriate reference to each other and how the communication lines should exist between them” (OGC, 2002) (itsmf.fi, 2009)

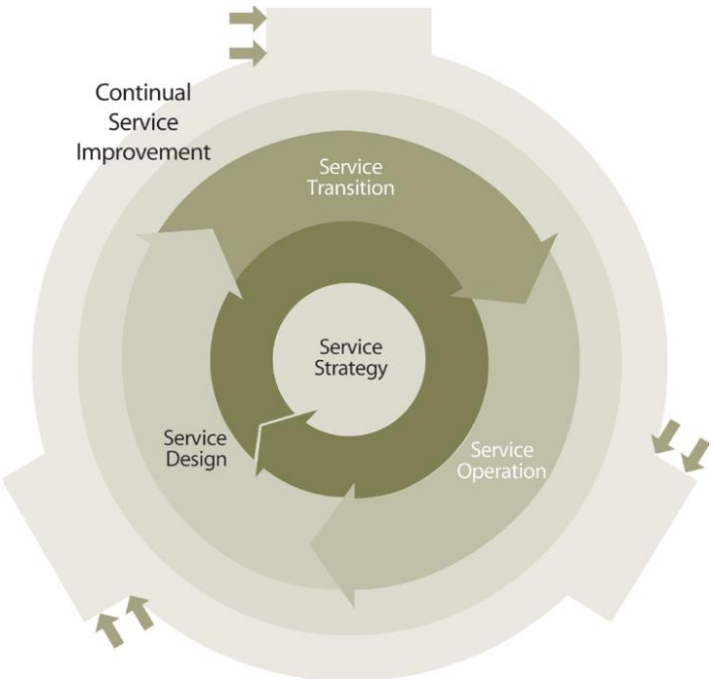

FIGURE 4. ITIL lifecycle model (OGC, 2007)

4.2 ITIL service life cycle

4.2.1 The Framework

The different stages of ITIL life cycle describe different processes and activities that the service can be managed with. The main concepts in the ITIL service lifecycle are the Service strategy, Service Design, Service Transition, Service Operation and Continual service improvement; these provide the framework of best practices, or a checklist to provide the service. The different phases of ITIL lifecycle are described briefly in the next chapter and the service operation is opened up in own chapter.

The ITIL uses functions, processes and activities to describe certain actions within each phase of a service's lifecycle. Within the ITIL the functions are defined to be a group of people or organizational units to carry out specific work and are responsible for the outcomes. These functions are self-contained and have the capabilities and resources necessary to perform and accomplish the outcomes. The capabilities include the work methods internal to the functions. The functions are used to structure the organization and define roles and associate authority and responsibility for outcomes. The process is defined in ITIL to be set of activities used to deliver a specific result or outcome. The characteristics of a process are: the process is measurable so the cost, quality and other variables can be validated, the process delivers a specific result, and the results are individually identifiable and countable. The processes deliver the results for the customer and in this case the customers may be either internal or external, the process responds to a specific event and the triggering event should be traceable; the previously described is illustrated in a figure 5. The activity is defined as a set of actions designed to achieve a particular result and the activity is usually documented in the procedure (document) that describes the steps to achieve the particular activity. (OGC, 2007)

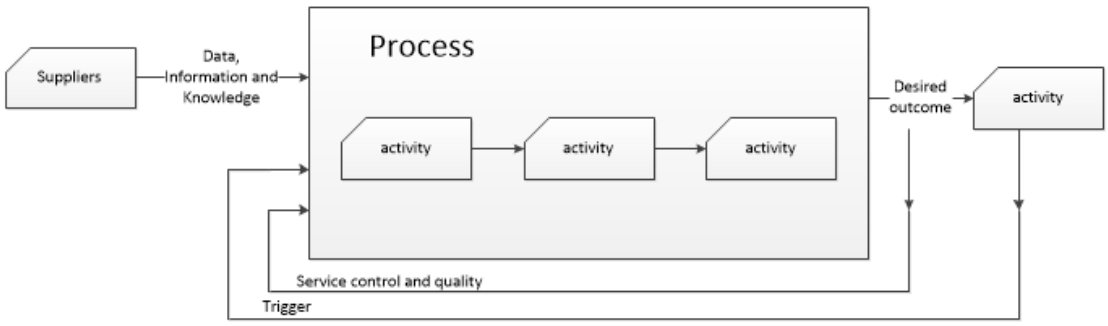

FIGURE 5. A basic process defined in ITIL (OCG, 2007)

4.2.2 Strategy, Design, Transition and continual service improvement

The service strategy is the core of the ITIL service life cycle and as a strategic resource it guides in the designing, improving and producing the service. Its goal is to form a plan or strategy how to serve the customers. The Service Strategy covers and explains the development of markets, internal and external service assets, service catalogue and implementation of strategy through the service lifecycle, financial management, service portfolio management, organizational development and strategic risks.

Service Design handles the designing and development of services and service management processes. The most important goal is to plan how the new or changed services are taken into use in production environment. The Service Design explains the design principles and methods for converting strategic objectives from the Service Strategy into portfolios of services and service assets. This phase also includes the changes and improvements necessary to increase or maintain the value to customers over the lifecycle of services, the continuity of services, achievement of service levels and conformance to standards and regulations.

The Service Transition consists of managing and coordinating the processes and systems that build, test and deploy the new and changed services. The

service transition creates the services defined in the service strategy aligned with customers and support groups. Efficient service strategy ensures that the new and changed services are better compatible with customers' business. The service transition includes practices in release management, program management and risk management, and places them in practical context of service management. It provides guidance on managing the complexity related to changes in services and service management processes preventing undesired consequences while allowing room for innovation.

Continual service improvement comes in when there are services that need to be continuously improved, as it happens to be today in many IT-departments. In this phase there is a need to be able to separate the profitable services from services that still need to be developed. Within separating these, the measuring and analyzing is a central theme. Continual service improvement measures and supervises that the processes are attended, checks that the processes fulfill the goals and see to that the process is efficient, and whether or not the processes provide extra value. The Continual service improvement combines principles, practices and methods from Quality Management, Change Management and Capability Management. The Plan-Do-Check-Act (PDCA) model specified in ISO/IEC 20000 is used at this phase. (OGC, 2007) (itsmf.fi, 2009)

4.2.3 Service Operation

Service Operation's function is to coordinate and produce the activities and processes that are needed to deliver and manage the services for business users and customers in a way that the delivery and management function on a level agreed on the service level. The service operation is also responsible for producing services and supports the management of required technologies.

Within the Service Operation there is the Event Management process that oversees all the events taking place in the infrastructure during normal usage

to measure the capacity. The Event management requires automated solution to track the anomalies and it can be used as a basis for automating, for example executing scripts based on the anomalies. The Event management can be applied to any aspect of service management that needs to be controlled and can be automated. The events in event management are basically just notifications of something that has happened and these events can be categorized.

Incident Management process concentrates on fixing errors on services as soon as possible in a way that it has the least interference with business and for the customer, thus ensuring that the SLA levels are maintained. Most of the incidents that occur are not new occurrences and they most likely involve something that has already happened before and might happen again, so there is need for predefined model to handle these issues

The Problem Management process includes root-cause analysis to resolve the issues behind the incidents, proactive methods to prevent incidents before the incidents can arise and a Know Error sub-process to resolve and diagnose incidents that might occur in the future. Also the Problem Management includes the tasks to minimize the impact of the incidents that cannot be prevented. This process maintains the knowledge about the problems and how to resolve or work around these problems. Access Management Process handles the access rights to decide whether the user has proper rights to use the service and denies access from those who do not have proper credentials.

The previously described processes do not themselves provide the effective Service Operation and there is a need for stable infrastructure and skilled people; the Service Operation relies on the groups of skilled people. The main groups defined in Service Operation are the Service Desk, which is the primary point of contact; Technical management provides the technical skills and resources to support the operation of the IT infrastructure; the Operations Management executes the daily operational activities to manage the IT

infrastructure; the Application Management is responsible to managing Applications throughout the application's lifecycle.

Some of the common Service Operation activities that are referred later at to this work are briefly opened up as follows (OGC, 2007):

The Monitoring and Control activity measures and controls the services and is fundamental to the delivery, support and improvement of services. The monitoring refers to actions that are made to observe the environment and to detect the changes that happen in the environment. The Control refers to the actions and processes that are made to manage the utilization and behavior of device, system or service.

Service Level Management process is responsible for negotiating the service level agreements, makes sure that these levels are met by monitoring these levels.

The Server- and Network - Management describes actions and procedures the server team makes to support, manage and measure the environment. Backup And Restore is described in the IT Operations and includes the checklist how to plan the backup and restore activities.

5 SERVICE AUTOMATION

5.1 Problem and Mission

When managing the infrastructure, there are a many different tasks that need to be repeated time after time. When working in small environments or the scale of services provided is small, the task repetition might not be an issue; however, when the scale rises the amount of work spent on these tasks might become overwhelmingly resource consuming. To ease up the infrastructure management and to free the resources to other tasks there is a need to automate some of the repeatedly done tasks that consumes the work

resources. When the automation is at work, the freed resources can be redirected to other relevant tasks and there is more room for improvement of the productivity. Although all these automatized solutions sound “god’s blessings to geeks”, these solutions need a significant amount of planning to be implemented, and when at use, these automatized solutions need to be supervised to verify the overall functionality of the solution. Figure 6. is somewhat of humorous figure, which still is a good figure to describe the issue at hand.

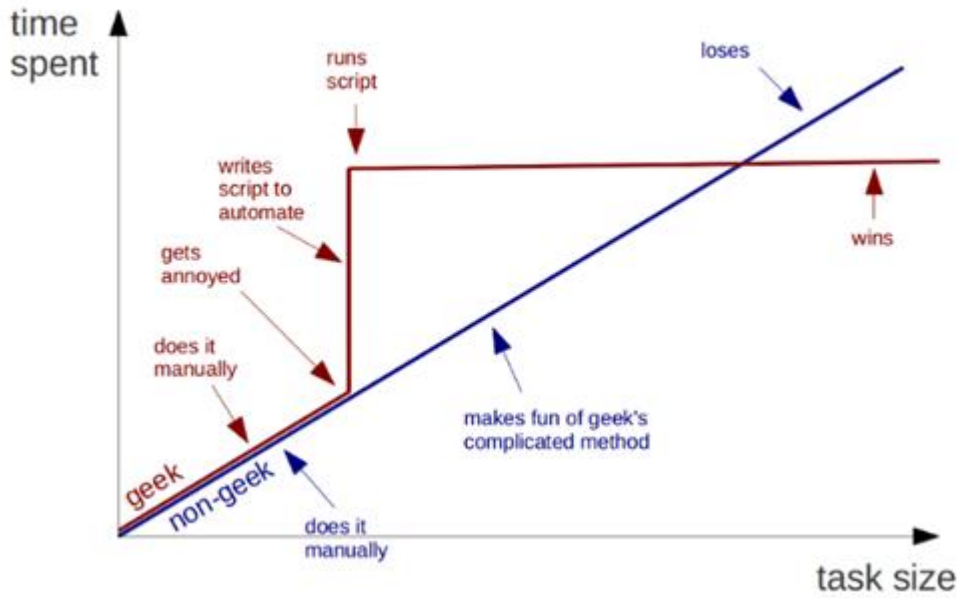

FIGURE 6. Resources vs. time (Johnson, 2012)

The mission is to ease up the management of launching the instance at OpenStack cloud environment. The work around this thesis starts with describing some scenarios and then moves to describing how the virtual computers can be manually started at JunkCloud environment. The manual chapter describes what are configurations required to launch the virtual computer are, what tools can be used in a case disaster, what the tools to locate the problem are and finally what tools are used when shutting down the virtual computer. After describing the manual steps, the work with forming up

the automatized solution begins using sequence flows and work flows based on the information from the manual work. The function of the sequence flow is to open up and describe the relationships between used services and machines to point out the possible need for some sort of API-solutions and where the automation solutions are needed. The work flow's function is to describe the process and functions/activities that are required to create the script. The work flows itself does not provide any solution, but it is used as a reference to find out the necessary tools with which the Python scripts can be made. When describing the manual steps, creating the sequence flows and creating the work flows, all these phases are divided into three different sub steps. These sub steps are "starting up", "running" and "terminating", which represent the services current phase. Finally, the tools with which the automation can be accomplished are represented in their designated chapter.

5.2 Scenarios

This chapter describes three different scenarios from the lifecycle of the services from the administrator's point of view. The scenarios describe the three main stages of service: the launching of service, the service upkeep, and shutting down the service. These scenarios are later used as a reference in this thesis.

The first scenario, launching the service:

The whole process starts when the cloud administrator receives a request from the service desk that describes the customer's need for a FreeNest service for his project group. The request also states that the administrator should inform the customer as soon as the service ready to be used and also all the required information regarding the usage of the service should be sent for the customer. The work that the administrator does to bring up the service is all done manually punching the commands step by step without any scripts that could help in this cause. First the administrator creates the networking related configurations to the environment for the service to be reached and be later usable. After the network is configured the virtual computer is booted up to the environment and the services are installed and configured within the

virtual computer. The steps required to launch the instance are described in the manual configuration examples.

The second scenario (upkeep of service):

As the service is running in the cloud environment, the administrator's tasks are to take the daily backups from the services and to resolve the disasters that might occur while the service is running at the environment. As in the first scenario, the work is mostly manual work and the incidents are resolved as they appear. In this scenario the incidents can be triggered not just by the administrator, but also by the customer through the service desk.

The third scenario (terminating the service):

As the customer sees that the service is no longer needed, the customer contacts the administrator through the service desk and informs about the request to terminate instance. The administrator makes the tasks required to terminate the instance manually using the command line and removes the instance and the data related to the instance from the infrastructure.

5.3 Manual configuration examples

5.3.1 Objective

This chapter describes the required steps and commands to start a virtual computer in JunkCloud environment. The steps to launch the instance are described and the variables that are used in the commands are also mentioned if those variables have an effect how the instance functions in the environment. The variables are listed for later reference and are needed when actually creating/writing the script; the creation of the script is not within the scope of this thesis. The work is carried out using fully functional OpenStack multi node environment. The manual configuration examples are divided into

three different chapters based of the instance's lifecycle; "starting up" representing the steps to boot the instance; "running" representing the phase when the instance is running in the environment; and "terminating" representing the steps when the instance is brought down. To keep the thesis little more fluent to read, the Running and Terminating phase can be found in Appendix 1. Manual configuration examples (Running) and Appendix 2. Manual Configuration examples (terminating).

5.3.2 Starting up the service

Based on the first scenario, this is the point where the administrator has already gotten the information to start the virtual computer through the service desk. As the administrator starts to work, the first task that needs to be performed, is to export project environment variables from the file called "novarc", thus making the euca2ools and/or OpenStack related commands functional. The novarc file is a key file to operate the OpenStack environment and the file itself is related to only one project. As the euca2ools commands require to be issued with sudo rights, the simplest way for the administrator is to change to "root" –user using the "*sudo –i*" command, and after changing to root user the environment variables are exported to system. Below is an example of used commands and it is done using the OpenStack nova-api machine:

```
$ sudo –i
$ ./root/creds/novarc
```

After exporting the environmental variables, it is time to start the instance. When starting the virtual computer, there are different variables that affect how the instance functions, for example how much memory is allocated for the virtual computer, additional virtual disk space, security group etc. At this point, there is base image(ami), kernel(aki) and ramdisk(ari) available in the environment that can used to start the instance. The image codes can be found using the "*euca-describe-images"* command and the instance can be booted using the "*euca-run-instance*" command. Below is an example of these

commands and the euca-run-instance command contains the minimum required information to start an instance; the minimum amount of information is the instance-type "–t" (memory allocation) and base image information. The mandatory and alternative variables of this command can be found in Appendix 3, list of variables.

```
$ euca-describe-images
                [Prints list of available images]
$ euca-run-instance -t m1.tiny ami-00000024 --kernel aki-00000012 \\
  -g server-group
```

The virtual computer might also require additional disk space for numerous reasons. The first step is to introduce this to the environment using the "euca-create-volume" command, the command creates the volume that can be distributed from nova-volume to instances. After the volume has been created, it needs to be attached for the instance using "euca-attach-volume" command. The "euca-attach-volume" requires few attributes to be used successfully and can also be seen in the example command below; the information required is instance to be attached to "-i", device name with which the instance sees the volume "-d", and the volume id that will be used.

```
$ euca-create-volume -s "size" -z nova
$ euca- attach-volume –i i-0000018 –d /dev/vdb vol-0000009
```

As all the traffic designated for the instance is blocked by default, the traffic needs be allowed for the instance to be able to listen incoming traffic. The traffic originating from outside source can be allowed by using "euca-authorize command". The instance can be allocated an additional IP address, public address, with which the instance can be reached from outside of the local area network. The address is firstly introduced using the "*euca-allocate-address*" command, and then the address attached to the instance using the "euca-associate-address" command. The example commands below allow

the SSH and ICMP-echo traffic to reach the instances that are in security group called “server-group” and gives the instance a public IP address.

```
$ euca-authorize –P tcp –p 22 server-group
$ euca-authorize –P icmp –t -1:-1 server-group
$ euca-allocate-address “some.public.ip.address”
$ euca-associate-address –i i-0000018 “some.public.ip.address”
```

When the instance has been launched, the state of the machine can be inspected using “euca-describe-instances” tool. When the state of the machine is “running”, it can be assumed that the instance has been booted up successfully and is ready to act as a virtual computer. The command can also be parsed using the piped grep command to get the relevant data out of the console output.

```
$ euca-descrbe-instances
$ euca-describe-instances | grep i-0000018
```

In JunkCloud environment, the connection from outside network can be allowed by adding port forwarding rules for the routing machine using iptables commands, the routing machine is a Ubuntu server that acts as a router for the environment. As an example command below, the traffic that has destination address 192.168.123.123 and port 2222 will be translated to be forwarded to address 10.0.11.45:22, that is the address of the instance located within the JunkCloud. The Second rule in the example is to allow the traffic to be forwarded that has come from interface eth1(WAN interface) and is destined to address 10.0.11.45 using port 22.

```
$ iptables –t nat –A PREROUTING –p tcp –i eth1 –d 192.168.123.123 –
dport 2222 –j DNAT –to 10.0.11.45:22
$ iptables –A FORWARD –p tcp –i eth1 –d 10.0.11.45 –dport 22 –j
ACCEPT
```

Using the information explained, the instance should be successfully running in the environment, but the instance still needs to be configured for it to deliver the requested services. The configuration of the FreeNest services within virtual computer is done through shell connection and is outside the scope of this thesis.

5.4 Work Flow

5.4.1 Objective

As some knowledge how to manage the infrastructure has been gained through manual work, the planning of upcoming system or scripts can be started. The first step is to abstract the previously used users and user interfaces as objects. These objects represent the “place” where the actions are performed and the relationships between these objects are clarified. As a result there will be sequence charts that visualize the relationships between these objects and the goal is to clarify the spots where some sort of API solution might be needed and to highlight where the automation occurs. After these sequence charts are done and described, the work flows will be made for all the three different stages of the instance’s life. The activities within the process will be made using the knowledge how the infrastructure was manually managed. As the activities for the work flows are done, the activities will then provide some technical problems and questions regarding what kind of solutions are needed to be solved for the scripts. The work flows will work as a frame for finding the tools with which the scripts can be made. At the end of this chapter the work should stand in a situation where there should be a lot of questions about how the functions in activities can be achieved; these problems are resolved later in thesis.

As in Manual configuration examples chapter, the Running and Terminating phases can be found in Appendix 4 and 5.

5.4.2 Starting up

The objective of this chapter is to create the work flow that describes the steps and activities necessary to start a virtual computer in JunkCloud environment. Based on the first scenario and the steps done on manual work, the virtual computer was brought up and configured using three different machines altogether. The machines that were necessary to bring up the virtual computer were as follows: a machine that was used to issue the nova/ec2 related commands to launch the instance, a machine that was used to configure networking; and the instance itself had to be configured for it to deliver the wanted services. As the launching of instance, configuring the instance and configuring networking were carried out using different machines, these are separated as different objects called nova, Instance and Networking for the use in sequence flow. The post configuration is made an own object to highlight the need for post configuration of instance and especially to highlight that there is a need to get the configuration information from somewhere in a way that there is at least user or administrator intervention. The infrastructure admin that was responsible for issuing the commands to launch the instance is now abstracted to be “Script” object and the same is done for the imaginary user that requested the instance. As there is always little waiting time when launching the instance because of the time that data takes to transfer from point A to B, there is a need for a “poller” object that checks when the instance is in a proper state to be delivered. The last thing that is needed is to come up with some sort of a logging object, because in a manual working environment the administrator is a well aware what is going on and on what phase the work is going on, however, when automating this information need to be available though some other channel. The objects and source of the objects are represented in figure 7.

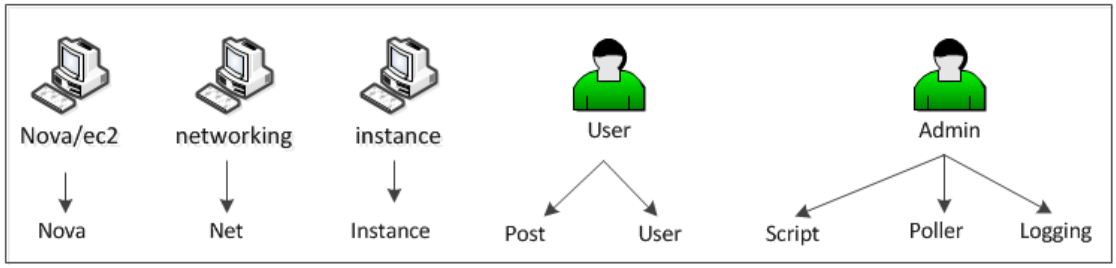

FIGURE 7. Objects for the starting up sequence flow.

Figure 8 represents the relationship between the objects for startup script and it is visualized in a timeline, as some events or actions between these objects require some event to be finished before the next object can operate.

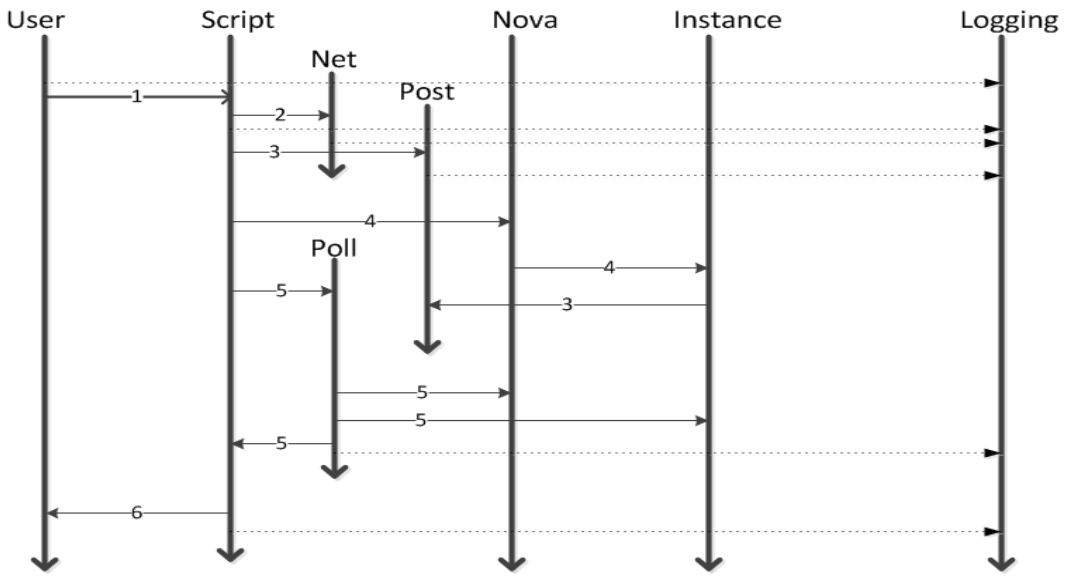

FIGURE 8. Sequence flow for staring script

The sequence flow in figure 8 is opened up as follows:

User & Script (1) – The first relationship is between the user object and script object where the user initiates the whole process to start the virtual computer and there is a need to eliminate the service desk personnel in the middle. This is the point where the first API solution is required, and also this is the start of the automation process.

Script & Net (2) – The script uses the Net object to configure the networking in a way that the network connectivity can be made to an

instance and the instance also has connectivity to outside world. The networking configuration cannot be issued to the same machine where the script is located so the relationship between these two objects needs to be accomplished in a way where the commands or configurations to configure network can be issued non-locally.

Script & Post (3) – Before the instance boots, the script uses the post object. The post object is used in a way that the instance can later obtain the configuration information from the location where this object has placed them. When the instance boots for the first time, it gets the required information from the post configuration database to configure itself and all the services that the machine needs to deliver. The relationship between these three objects, the script, post and instance most likely need some sort of solution that cannot be made just using one machine.

Script & Nova (4) – The script handles the tasks required to launch the instance and it uses the nova object to accomplish the instance launching. The relationship between these two might need some sort of API solution if for some reason the nova/ec2 related commands are not used and the script itself is not, and most likely will not be located in the same machine where the nova-api is located.

Script & Poller (5) – The second last object that the control script uses is the poller. The poller's function is to check when the launched instance becomes up and ready to be delivered. This information can be obtained from the nova object and making some sort of a reachability test for the instance. When manually booting up a virtual computer, there is always a little wait time before the virtual computer is up and running, thus the instance cannot be delivered right after the instance launching command is issued.

Script & Deliver (6) – After the controlling script gets a verification from poller, the script relays the mandatory information and credentials for the end user through the same media as the request came through.

Logging (*) – All of the objects push logging data for the logger object. This data can then be used for error management, measurement and to

trigger events. The logger object is responsible for handling the messages that the objects send. The relationship between the logger and other objects requires a way for the objects to send the logging data for the logger.

Using the sequence chart that visualizes the relationships between interfaces and knowledge how the instances were manually launched, the activities for the work flow can be made. The result of this will be represented in figure 9 and the activities within the figure are opened up and described activity at a time. Within the descriptions there are references to ITIL processes and activities that can be used to support the activities in the process of starting the new service to environment or what activities can be used in the work flow to support other ITIL processes and activities. As the activities in the work flow have more or less relationships in different processes and activities described in ITIL, there is only a short notification about with which process, activity or function the work flow has a relationship with. The internal state machine in this workflow follows the path of the activities and the state of the system changes based on the current activity.

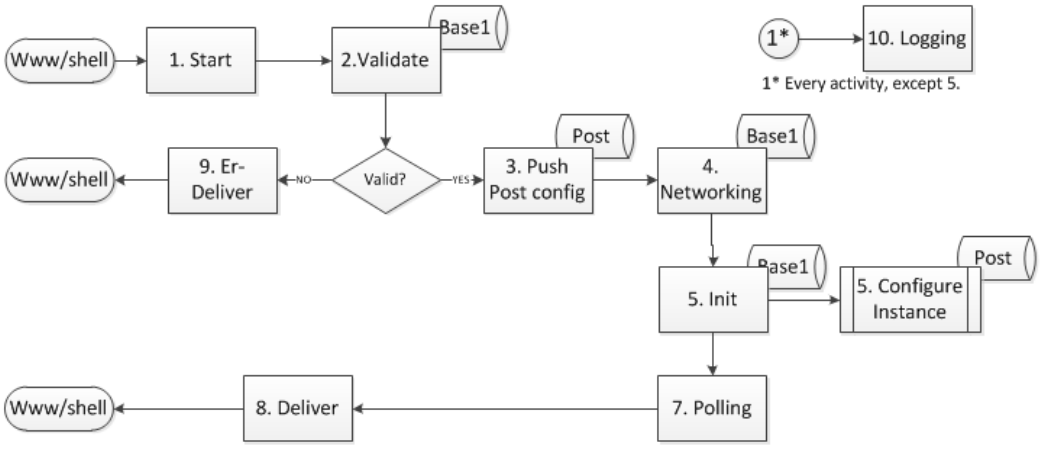

FIGURE 9. Work flow for starting script

Activities within figure 9 are described as follows:

Start Activity (1) – This is the first activity that initiates the whole process to start the instance. This activity listens for incoming triggers that come from either web or shell terminal. Start activity represents the situation where the administrator receives the order to start a virtual computer to the environment. When referencing to ITIL, this can represent the situation where the Service Desk has gotten the request to deliver the new service and the request will be forwarded to Server Management that will handle the process to start the service to system.

Validate Activity (2) – The next activity the administrator has to do, is to check that he had all the required information required to start the instance and the information gotten is in correct form. Validate activity's function is to make different validation checks for the request to decide whether the instance can be started or not. The check can be made cross referencing the information in database "base1" containing the information what kind of request can pass the sanity check. When referencing to ITIL, this activity needs to use the information from the Access Management process to determine whether the user has proper rights to start the service.

Push Post Config Activity (3) – This activity pushes the information or configurations to "post" database where the instance can get the configuration information required to complete its software installation and configurations as post-install. When referencing to the scenario 1, this activity represents the information that can be gotten from the request that were designated for the administrator and represents the need for the services and configurations that the instance must deliver.

Networking Activity (4) – After pushing the information to post database, the networking related configurations are to be done, so the instance to gain network connectivity. The relevant information about the configurations made will be placed on database base1, as this information will be later needed on different activities. This activity represents the actions that the administrator must do for the instance

gain network connectivity. To reference the ITIL, this activity can be made using the help of Network Management described in ITIL.

Init Activity (5) – The Init activity is responsible for starting the instance in the infrastructure using the information originated from the request and information from the database “base1”. This activity represents the actions that must be made to launch the instance to infrastructure.

Configure Instance Activity (6) – After the instance has booted, it does not mean that the instance is up and ready to deliver the wanted services; the instance still requires the post install and configurations to be completed. The Configure Instance activity handles the post configuration of the instance using the information that can be found in the location where the (3)Push Post Config activity placed this information. This represents the configurations that the administrator must make for the instance to deliver the services. ITIL reference here is the Server Management and Support thus the work is made by the group whose responsibility is to manage the servers.

Polling Activity (7) – As the instance is booted, there is always little wait time as the infrastructure moves instance data to the compute-node. The poller activity checks from the infrastructure when the instance becomes up and active. Once the instance is up, the work can move to next activity.

Deliver Activity (8) – After the instance has been brought up and is configured, the information how an end user can connect to an instance can be delivered for the end user using the same channel where the request originally came from. This activity represents when the administrator informs the user about the usage of instance in first scenario. Referencing to ITIL, as the Server Management has finished the instance configuration, the tasks to deliver the instance related information are delegated for the Service Desk.

Er-Deliver Activity (9) – In a case the request does not pass the sanity check, the instance launching the process is terminated and the end user is informed about this incident. Referencing to ITIL, in a case there

are some errors in some stage of the process, the customer can be informed through Service Desk.

Logging Activity (10) – All the activities send information to Logging activity. This activity handles the information based on the type of the message received and triggers events based on the received message. Referencing to ITIL, everything that can be monitored can be sent through Event Management for tracking and measurement. In a case of errors, the Even Management makes appropriate actions based on the triggering error. .

5.5 Tools for Work Flow

5.5.1 Objective

At this point there is a clear vision of what is wanted to be achieved, a script that simplifies the launching of the instance, creates backups, tries to resolve disasters, and simplifies the terminating of the instance. The activities these scripts keep within are described using the work flows, however what is not yet clarified are, the tools with which these activities can be made to function. The goal of this chapter is to find the tools with which the activities can be accomplished. When discovering the tools, it has to be kept in mind that these tools have to be someway compatible with Python scripting language that is used to glue these pieces together.

As in the previous chapters, the work is structured to three different sections based on what phase the instance is in; these phases are the starting up, running and terminating. The activities in work flows are opened up one activity at a time and the required tools are described for this activity. The tools and solutions represented might not, and most likely are not the most optimal or the best solution for the represented issue. The research for the most optimal tools is left outside the scope of this thesis.

The recovery, backup, terminating and logging chapters can be found in Appendices 6 – 9.

5.5.2 Starting up

The previously represented figure 9. Work flow for starting script can be found in chapter 5.3.2 and is used as a reference point to find the required tools to accomplish the described activities within the work flow. The activities described previously are opened up activity at a time and the tools to accomplish the functions in activities are represented.

The first activity in figure 9, start activity (1), is responsible for listening the incoming triggers and initiating the process; the goal is to find out how the system can gain information originated in the web terminal. Running the script through command terminal is not an issue, but the initiation of the script through web is something that is preferably wanted for the system. The solution for this problem can be achieved by using restful solution creating client-server relationship between the web terminal and the initiation script, where the web terminal acts as a client for the service (WSGI, n.d). The activity also needs to parse the JSON or XML data that comes within the trigger for the next activities to properly function (Python Software Foundation, n.d). Figure 10 summarizes the described solutions for this activity.

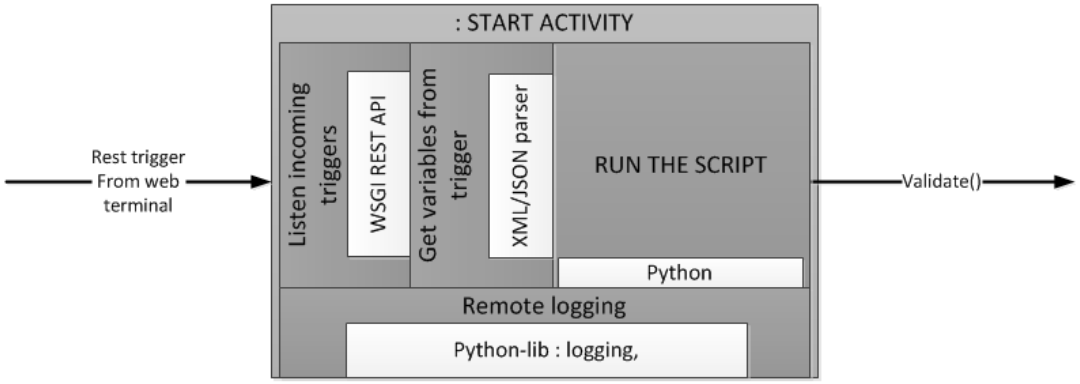

FIGURE 10. Proposed solution for activity 1. in the startup script

As the trigger to initiate an instance has been received, the validation activity (2) makes comparisons between the data parsed from the trigger and the data in the database. The comparison is conducted to check if the user has the rights to start the instance and the information in the trigger is in the correct form. These sanity checks can be done with simple comparisons between the data within the request and data in database. Additionally the data must also be in correct form for the request to be passed on to the next activity to minimize the possible problems from misconfiguration and to prevent abusive usage using altered messages. In a case where this received information does not pass the sanity check, the received event will be dropped and information of this occurrence will be sent to logging activity. Within the script, the database connection can be made using python sqlite3 library (Python Software Foundation, n.d). Figure 11 summarizes the described solutions for this activity.

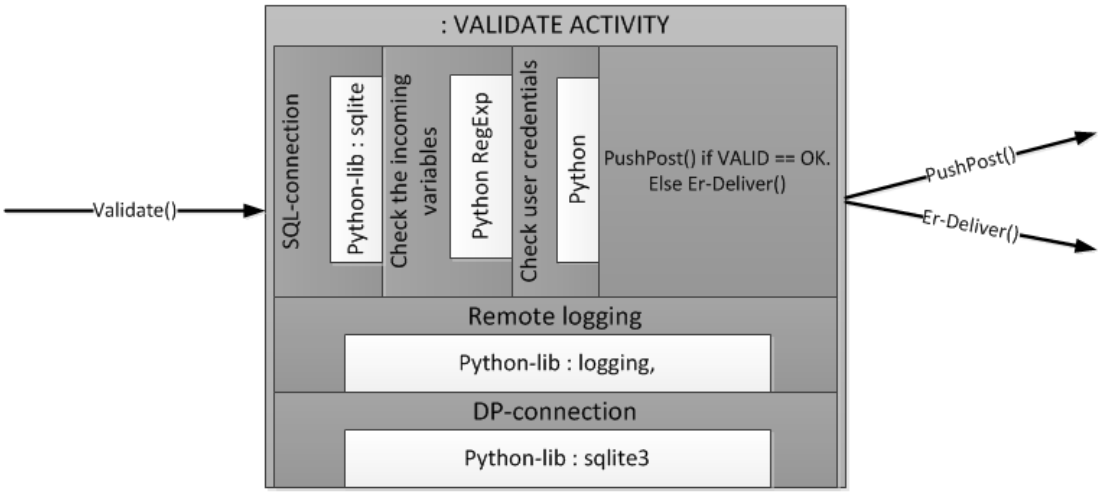

FIGURE 11. Proposed solution for activity 2. in the startup script

Pushing information to post configuration database is done by Push Post Config activity (3), and the activity is responsible for accomplishing the base work that is required to get the configuration information for the later use in instance configuration. The goal is to find a way to automate the instance configuration. This activity can be made using cloud-init compatible startup script that is also introduced in OpenStack documentation. Within the activity, the script is first made and then it is located so it can be later used when required. The cloud-init is a package with which configurations can be made to the instance during the instance's first boot (Ubuntu, n.d). Within the cloud-init there is one function that comes handy when creating something that has to be achieved as a post install, the User-Data scripts. This User-Data script is run at the later phase of the first boot of instance and it is written as normal shell or Python script. By using this script the instance can be configured to be exactly as it is wanted to be. The downside of the cloud-init tool is that it is only usable on Debian based operating systems. For other operating systems, there are other solutions to run the startup scripts, however, there are no proper tools that would work on every operating system as most of these tools are specific to some operating systems only. Figure 12 summarizes the described solutions for this activity.

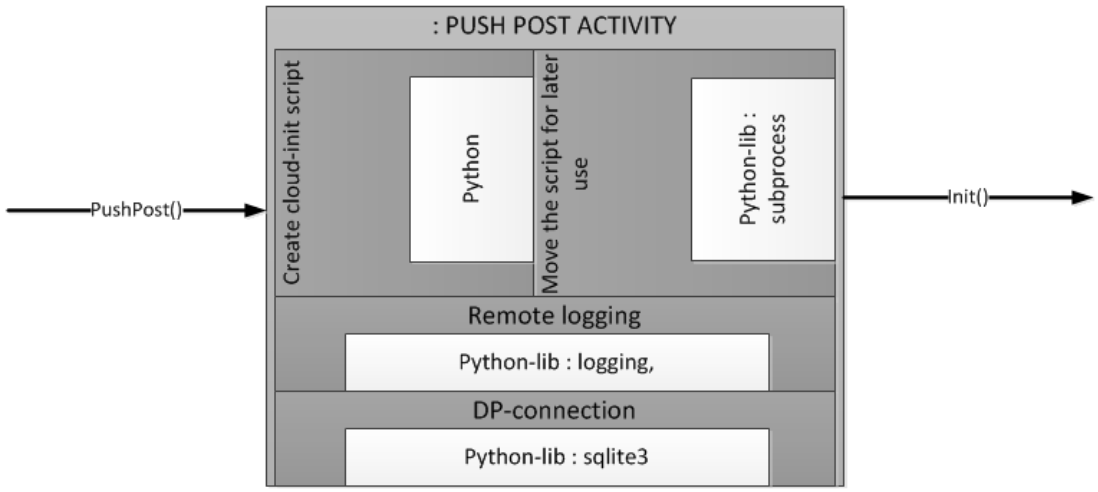

FIGURE 12. Proposed solution for activity 3. in the startup script

The Init activity (5) is one of the activities that have the most relevant functionality for the startup phase, so a proper tool must be found to orchestrate the OpenStack infrastructure using scripts. Whatever the tool is, the goal is to start the instance using the information that has been gained through previous activity.

To accomplish this goal, there are three different solutions that can be used. The first is to issue the launch of an instance from local machine where the script is located, and this requires the Ubuntu Nova tools to be installed. The second way is to send restful messages to nova-api machine using either curl or pythonic nova commands. The third solution is to find a tool with which the commands can be issued to remote machine. The second solution would be the most laborious to make to work; also as the OpenStack is still under development, the way the restful messages are handled and sent might change and it might bring some unwanted problems and changes to script. In this case, the simplest way would be to use the local nova or euca2ool tools to launch the instance and it would mean that the internals of the tool can be changed without changing the functionality of the script. The downside of using the nova/euca2ool is that the credentials to handle the cloud infrastructure are located in the local machine and it could bring some unwanted problems that the author of this thesis is at the moment aware of. The third solution requires some tool to be used to orchestrate the cloud or

remote machines. Based on Ossi Rantapuska's thesis, Configuration management of FreeNest (Rantapuska), the tool that could be used to orchestrate the system is called Fabric. This tool comes into play because with it the terminal commands can be issued to local or remote locations. Thus, the fabric can be used as glue for the system to overcome these issues, basically, if things can be done through command line, it can be scripted and issued using fabric. The way the OpenStack glues pieces together is to use messaging queue technology, RabbitMQ for example. Figure 13 summarizes the described solutions for this activity.

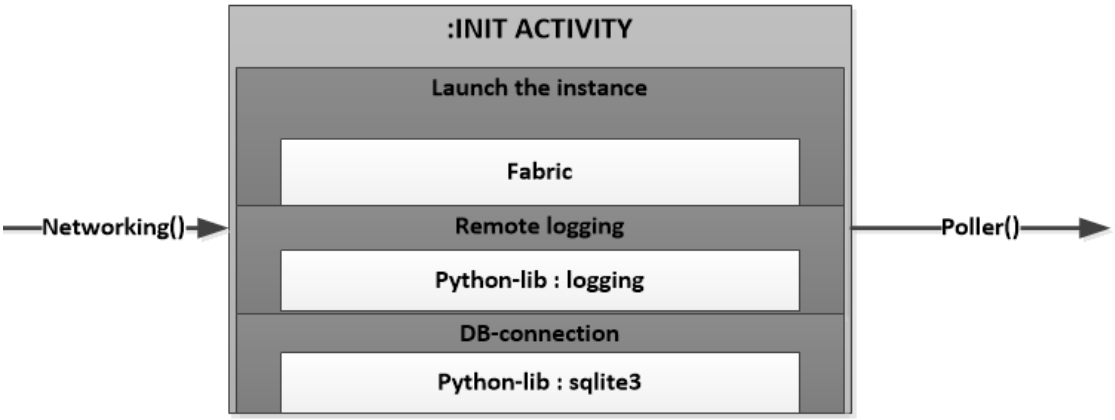

FIGURE 13. Proposed solution for activity 5. in the startup script

Networking related activities (4) were carried out using shell connection to the remote machine, therefore here the Fabric comes in play, as the networking related commands are something that can be easily scripted. There is also a new OpenStack project called Quantum that aims to provide "network connectivity as a service". In short, the Quantum provides API to manage networking connectivity and uses plugins to manage the virtual and physical switches and also virtual interfaces related to OpenStack environment (OpenStack project, n.d). As the Quantum is still under development at Essex release, it is only recommended to be used at development environments and it is not used in this solution, although definitely a tool to look for in the future. Figure 14 summarizes the described solutions for this activity.

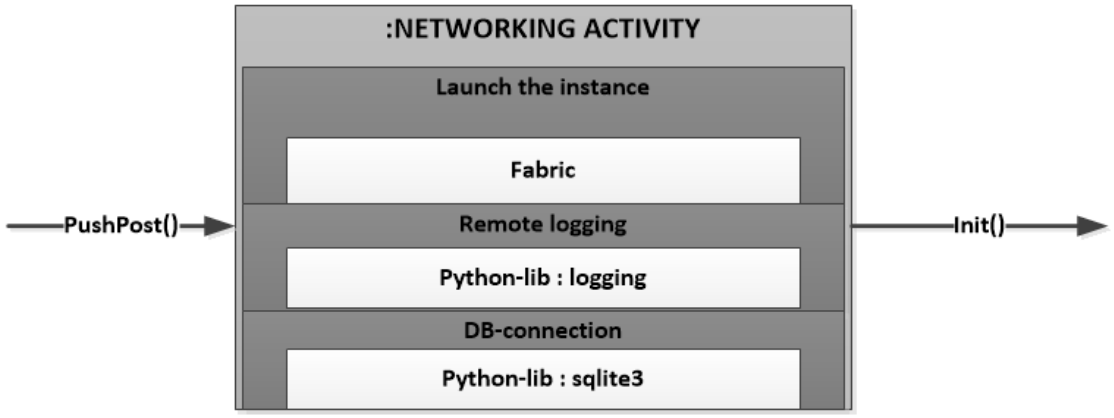

FIGURE 14. Proposed solution for activity 4. in the startup script

The instance configure activity (5) has become an obsolete activity and should not require any special activity as the goal is that the instance gains the possible configuration information with the help of cloud-init.

For the polling activity (7) the script checks when the launched instance is ready to be delivered. The problem here is to find a way to make sure when the instance is up and ready to be delivered. Manually it can be checked using the ping tool and checking from the nova-api. The solution will be achieved using Fabric to check the infrastructure when the instance has been brought up and checking that the instance is reachable using ping, which also can be issued using fabric. Figure 15 summarizes the described solutions for this activity.

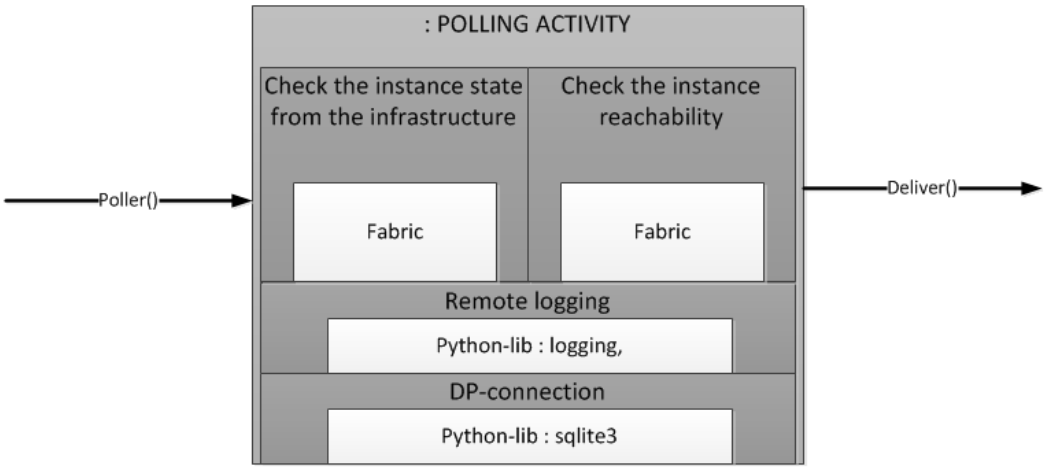

FIGURE 15. Proposed solution for activity 7. in the startup script

The delivery activity (8) places the information targeted for the end user in to a location where the web terminal can get this information and deliver it for the end user as dynamic content. The problem is to find a way to deliver the information from the infrastructure to the web server. The simplest solution is to locate the information to database where the web terminal can retrieve this information and represent it for customer. Figure 16 summarizes the described solutions for this activity.

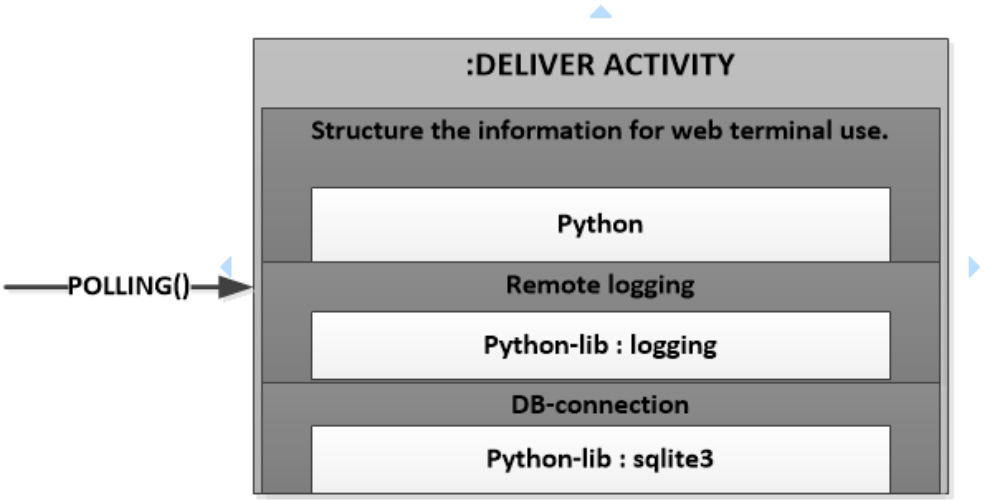

FIGURE 16. Proposed solution for activity 8. in the startup script

The logging activity (10) can be found in Appendix 9 where it is opened up in more detail.

6 CONCLUSION

The goal of this thesis was to find the tools to automate the lifecycle of instance at OpenStack based JunkCloud environment and the results of this thesis are the suggested tools described in the chapter 5.5.1 and in

Appendices 6 – 9. The work was done structuring the steps from the manual work to activities that were used in work flow to describe the whole process. The activities were then used as a reference to find out the required tools with which the described activities could be made in automated manner. In the end, there are suggestions of tools with which the proposed solution can be made, but the tools suggested are just one solution among other possible solutions and can be later used as a starting point if there is a need to find the most optimal tool for the current issue at hand. The results of this thesis don't provide conclusive information how to build the complete solution to automate the service lifecycle at the JunkCloud environment, but it provides the suggestion of tools with which the solution can be made and can be used as a starting point for future work to support the process to decide what tools to be used.

The other task of the work was to pointing out the ITIL processes and activities that could be used to support the automation process and vice versa. Pointing out the ITIL processes and activities was done when creating the work flows and describing the activities for the automated processes; references for ITIL processes can be found in chapter 5.4.2, Appendix 4 and Appendix 5. The ITIL referencing ended up being sort of a notification to point out the names of the ITIL processes that can be used to support the activities in the work flow and the detailed ITIL process descriptions were left outside of this thesis. The more detailed use of ITIL in this work was left outside as the ITIL requires much information to be clarified and explained about the processes and activities before it starts to make much sense; in the case of this thesis it would have bloated this work to cover issues enough for another thesis. The more detailed use of ITIL at the FreeNest service could easily give coverage for another thesis. Also it feels that it is somewhat hard to find exact answers from the ITIL, as it is a framework for service providing, it does not tell "it has to be done exactly like this", the answers are more likely a checklist of issues that need to be considered to deliver a healthy service.

In principle, this thesis followed a structure where the big question was pieced into smaller questions and again the questions were pieced into more smaller questions, until the questions were specific enough to be answered with something other than more questions. That is because the service is a sum of many components, and the deeper the system is inspected the more of these smaller tools are found that create the big picture, and that is how the answer for the big question can be found. To open up the big question, the work flow approach was used in this thesis and it seemed like a fluent way to go, at least for the author of this thesis; and by doing so it was possible to point out the ITIL processes at the same time while describing the activities in the work flow. The work flow used in this thesis is highly ITIL influenced due to the author's previous service management classes and lack of programming classes where the "proper" way to approach these things would have been gained, thus it was easy for the author to use this approach. For the people with more programming experience, the workflow approach might not open up the same way as for example describing schematics using the UML.

The biggest problem in this thesis for the author was to figure out how much the subjects explained can be cropped to keep the background information informational enough. There are a lot of subjects in this thesis that need more clarification, and if done so in this thesis it would have bloated the thesis too much for some of these subjects are broad enough to cover another thesis. The possible future research topics found in this research are: more detailed usage of ITIL in consept of FreeNest; REST WSGI solution with which the connection can be made between the infrastructure and web interface; how RabbitMQ can be used as glue to bind different systems together and what kind of problems can be solved with it in cloud infrastructure environment; alerting administrative personnel through SMS-system; automated configuration of launched instance using cloud-init package; research on OpenStack Quantum to manage networking configurations.

The future of cloud computing opens up a lot of questions about information security and these issues cause "grey hairs" to people who are concerned in this area of expertise, as it is not enough just to guarantee the data safety in the local premises, but also the safety of data transfer between the local

premises and cloud needs to be ensured. For the groups that find the lack of security inadequate, there is always a possibility to create private clouds to keep the full control in their own hands; also the management of the environment becomes easier as time goes by because of improvements in automation and this eases up the management of the environment. The openness in the internet's data traffic is one of the key concepts in making cloud service providing worthwhile and as long as it stays this way, cloud service will remain a growing area. The future of OpenStack seems bright as there are numerous "big names" collaborating to develop this toolset and it is open, thus making it a worthwhile solution to be implemented. The deployment of OpenStack becomes easier as time goes by and soon it might even be possible to initialize your own private cloud with just a "single click of a button".

The future where FreeNest service can be provided from the cloud gives the advantage of speed how fast the service is provisioned for the new user and in the future these issues become more essential. Also in the concept of FreeNest, the cloud opens other possibilities of what services/tools can be integrated to FreeNest service, for example automated service testing environment that can be launched through FreeNest service. For the future of JunkCloud, the author hopes that the environment stays as a testing environment for the sake of the project members, and thus giving them the freedom to test and play around in the environment and thus providing an environment in which to improve themselves without the risk of any monetary losses.

REFERENCES

itsmf.fi. 2009. *ITIL v3 Taskukirja.* s.l. : Van Haren Publishing, 2009.

JAMK ICT. n.d. Projektit - Jyväskylän ammattikorkeakoulu. n.d. Cited: 10 15, 2012. http://www.jamk.fi/projektit/1233.

OGC. 2007. *ITIL Service Operation.* s.l. : The Stationery Office, 2007.

OGC. 2002. *ITIL Service Support.* s.l. : Office of Government Commerce, 2002.

OpenStack project. n.d. *Official OpenStack site.* n.d. Cited: 7 7, 2012. http://www.openstack.org/.

OpenStack Project. n.d. OpenStack Compute Administration Manual. n.d. Cited: 7 15, 2012. http://docs.openstack.org/essex/openstack-compute/admin/content/users-and-projects.html.

OpenStack project. n.d. OpenStack Quantum administration guide. n.d. Cited: 8 19, 2012. http://docs.openstack.org/trunk/openstack-network/admin/content/WhatIsQuantum.html.

Pepple, Ken. 2011. OpenStack Nova Arhitechture. April 22, 2011. Cited: 11 4, 2012. http://ken.pepple.info/openstack/2011/04/22/openstack-nova-architecture/.

Python Software Foundation. n.d. Python Documentation. n.d. Cited: 10 7, 2012. http://docs.python.org/library.

Rantapuska, Ossi. *Configuration management of FreeNEST.*

theResearchpedia. n.d. *theResearchpedia - What is infrastructure as a service (Iaas).* n.d. Cited: 8 15, 2012. http://www.theresearchpedia.com/research-articles/what-is-infrastructure-as-a-service-iaas.

TIVIT - Cloud Software Finland. n.d. Cloud Software Program. n.d. Cited: 9 15, 2012. http://www.cloudsoftwareprogram.org/cloud-software-program.

Ubuntu. n.d. CloudInit Community Ubuntu Documentation. n.d. Cited: 8 12, 2012. https://help.ubuntu.com/community/CloudInit.

Wikipedia - Cloud Computing. n.d. Cloud Computing - Wikipedia. n.d. Cited: 15 6, 2012. http://en.wikipedia.org/wiki/Infrastructure_as_a_service.

WSGI. n.d. WSGI - WSGI.org. n.d. Cited: 16. 11 2012. http://wsgi.readthedocs.org/en/latest/.

APPENDICES

Appendix 1. Manual configuration examples (Running)

Based on the first scenario, as the instance is successfully running, there is a situation where the administrator needs to make backups from the running instance. These backups can be made taking snapshots from the running instance for backup purposes. The snapshot can be taken using the “nova image-create” command and the volume information can be snapshotted using the “euca-create-snapshot” command.

```
$ nova image-create [instance-id] [name]
$ euca-create-snapshot “volume-id”
```

Based on the first scenario, the administrator does not use any automatized tools to check if the instance is alive and running, so the admin has to do this manually or to start checking the problem if s/he gets information that the service is unreachable. There are various tools and methods to check if the instance is up and running; and the functionality of instance also relies on other parts of the infrastructure to work, not just the nova-compute component where the instance is located. The first thing there is, is a need to find out what part or component of the infrastructure has failed, so the recovery can be made without affecting other parts of the infrastructure. The most common problems solving tools that have been used to find the source of problem are described next.

The ping command line tool is very useful to find out if the network between two hosts is functional and it has end to end connectivity. As this tool mainly just gives information about the end to end connectivity, it is usually the first tool that is used to check the connectivity. The con of this tool is that it does not provide any information if there is a problem in between the end devices, so from this tool’s point of view if there is a problem in between the devices or in the end device, it always gives the same result.

$ ping target.host.ip.address

Euca2ools has command “euca-describe-instances” that gives the list of all the instances running in the environment. The tool gives the information whether the instance is running or not and it should be described to be in running state. If the instance is described to be in some other state than running, it means that during the launching of the instance something has gone wrong or the instance data has not yet been transferred to nova-compute machine.

$ euca-describe-instances
[LIST OF ALL INSTANCES AND INFORMATION]
$ euca-describe-instances | grep “ip.address.of.instance”
[INSTANCE INFORMATION]

The information whether the instance data transfer has been started or not can be checked from the logs of glance and the logs of target nova-compute where the instance is supposed to be booted.

The booting up of instance requires the other parts of the infrastructure to work properly, so when booting up the instance for a first time, the nova-api machine must know where the other components of OpenStack are located. This can be checked using the “nova-manage service list” command that prints information of whereabouts of the components. If the list is empty, or it does not contain the required services, it means that the required service has failed to register itself to the database and has been misconfigured. The list should include at least nova-network, nova-scheduler, nova-compute and nova-volume.

$ nova-manage service list
[LIST OF AVAILABLE SERVICES]

Sometimes the single component can crash and other services try to use it because the stateless nature of the services. To check that the service is running in the host machine the “ps aux” tool can be used and the location of service can be checked in the “nova-manage service list”.

```
$ nova-manage service list
$ ssh serviceX
$ ps aux |grep nova
```

The instance that has been suspended by some reason can be brought up by using “euca-restart-instance” command. The command tries to reboot or start the instance in its current location.

```
$ euca-describe-instance | grep i-12345678
        [instance i-12345678 state shutdown]
$ euca-restart-instance i-12345678
```

If the instance still fails to change it state to running state, the instance’s console output can be checked using “euca-get-console-output”. The command displays information about the instance’s boot process.

```
$ euca-get-console-output i-12345678
        [INSTANCE CONSOLE OUTPUT]
```

In a case the host machine is not responsive, the situation might require that the instance will be booted using the latest snapshots. In this case, the public IP must be removed from the non-responsive instance and allocated to for the new instance. The booting of the new instance is done the same way as described in the chapter 5.3.1.

Appendix 2. Manual Configuration examples (terminating)

The tasks required to terminate the instance are relatively straight forwarded when it comes to destroying all the data related to instance. In a case, where the instance data is required to be kept, it requires the snapshot to be made before the instance is terminated from the system. There might also be a situation where the instance is just wanted to be suspended rather than be terminated; if the instance is suspended, the data is not removed from the host nova-compute machine.

If the instance is just wanted to be terminated and the data is wanted to be removed from the host machine, the “euca-terminate-instance” command does this trick.

```
$ euca-terminate-instance i-12345678
```

If the instance is wanted to be suspended, the “euca-stop-instances” command can be used to achieve this. This way the instance data will not be removed from the nova-compute machine.

```
$ euca-stop-instance i-12345678
```

The instance data can be transferred to long term storage by taking snapshot from the instance that is either running or stopped. The snapshot can be taken using “nova image-create” command and the mounted volume can be snapshotted using “euca-create-snapshot” command.

```
$ nova image-create [instance-id] [name]
$ euca-create-snapshot “volume-id”
```

Appendix 3. List of variables

List of variables that can be used in commands found in the manual configuration examples chapter:

euca-run-instance

Variable	**Explanation**
-t m1.example, - -instance-type	Amount of virtual memory allocated for the instance, varies from m1.tiny(256MB) to m1.xlarge(16GB). Mandatory
ami-12345678	HD-image for the instance. Contains the operating system. Mandatory.
--kernel aki-12345678	Kernel image for the virtual machine (usually when published kernel is attached to certain ami)
--ramdisk ari-12345678	RAM disk image for the virtual machine.
-n “amount”	How many instances to start with same attributes
-g “group”	To what security group the instance will be allocated
-k	rsa public key that will be transferred to instance for later SSH connection
-d	Custom user data, used to give instance custom startup scripts.
--addressing	Addressing mode, public or private IP
-z –availability-zone	Security zone that influences the instance.
- block-device-mapping	Attach Elastic Block Store or instance store

	volume to instance.
-s – secret-key	User’s secret key

Appendix 4. Creating Work Flow (Running)

The running phase includes two different processes. One is responsible for creating backups from the instances, and the other is for polling the instances and components to check reachability. The second is also responsible for trying to recover from the disasters. The goal here is to create the work flows and describe the activities within these two work flows that are necessary to accomplish the processes.

As in the previous chapter, first the sequence flow is made. From the administrators actions the next objects can be derived for the first function; the objects are script, poller, recovery and logging. The “script” object represents the administrator that initiates the reachability tests and starts to act accordingly if the reachability tests indicate so. The “poller” object comes from the real work where the administrator manually makes the instance reachability tests and checks that all important services are alive and reachable. Recovery object is derived from the tasks what the administrator makes to recover crashed instance or service that has failed infrastructure. The used systems in this case are the instance, infrastructure and nova; which are described in sequence flow as different objects. The objects and source of the objects are represented in figure 17.

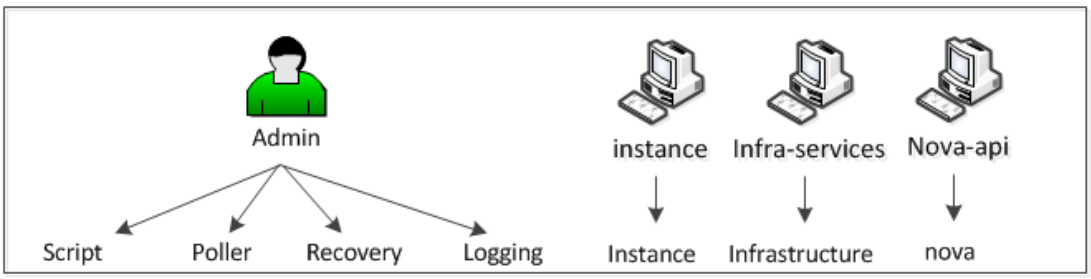

FIGURE 17. Objects for the recovery sequence flow.

Figure 18 represents the relationship between the objects for recovery script and is visualized in a timeline, as some events or actions between these objects require some event to be finished before the next object can operate.

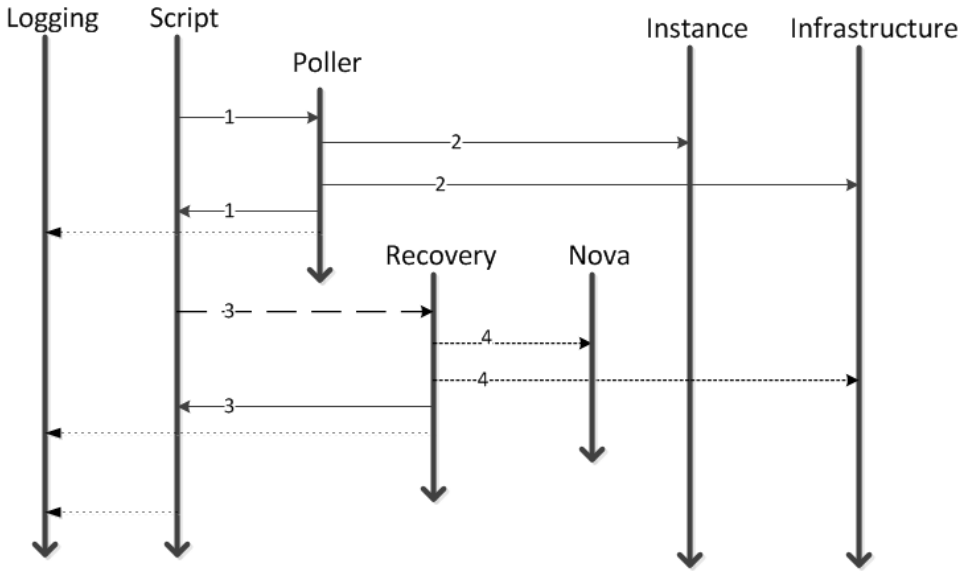

FIGURE 18. Sequence flow for recovery script.

The sequence flow in Figure 18 is opened up as follows:

Script & Poller (1) – The controlling script initiates the polling object that in term returns information whether the reachability test was successful or not.

Poller (2) – The poller makes the reachability tests to instances and to specified parts of infrastructure. The relationship here is poller's requirement to know where the services are located and what services are required to be polled.

Script & Recovery (3) – Recovery object is initiated if the poller returns information for the controlling script that states that instance or service is found to be unreachable. As recovery attempts have been made, the recovery object returns information whether the recovery has been successful or not.

Recovery (4) – This object attempts to recover the instance or service based on the information gotten from the controlling script.

Logging (*) – All of the objects push logging data for the logger object. This data can then be used for error management, measurement and to trigger events. The logger object is responsible for handling the messages that other objects send for it. The relationship between the logger and other objects require some way for the objects to send the logging data for the logger.

Using the sequence chart that visualizes the relationships between interfaces and knowledge how the instances were manually launched, the activities for the work flow can be made. The results are represented in Figure 19, and the activities in the figure are opened up and described activity at a time. Within the descriptions there are references to ITIL processes and activities that can be used to support the activities in the process of starting the new service to environment or what activities can be used in the work flow to support other ITIL processes and activities. As the activities in the work flow have more or less relationships in different processes and activities described in ITIL, there is only a short notification about with which process, activity or function the work flow has a relationship with.

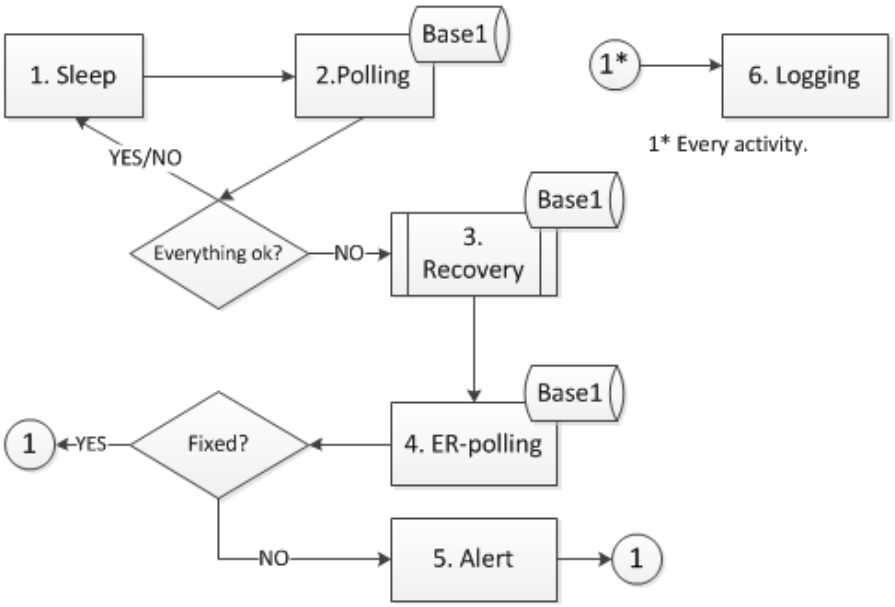

FIGURE 19. Instances disaster recovery work flow

Activities within the recovery work flow are opened up as follows:

Sleep Activity (1) – The sleep activity is something that is required for the system not to trigger polling activities too often as this loop would otherwise result in too much consumption of the server's resources. As the idle time has been passed, the sleep activity triggers the polling activity. ITIL reference for this activity the daily tasks described in the IT Operations what describes the Job Scheduling.

Polling Activity (2) – The Polling activity handles the reachability test made for instances and services. The Polling activity checks the database for what instances and services are required to be tested for reachability and where these are located. In a case this activity finds some service or instance to be unreachable, the recovery activity is called. This activity represents the activities that the administrator must make to check whether the services are reachable or not. The administrator must also locate the source of the problem so it can be acted accordingly. Referencing to ITIL, this activity can be made with the help of Monitoring and Control activity.

Recovery Activity (3) – The Recover activity is responsible for trying to resolve the instance or service disasters based on information that is received from the polling activity and information on database. This activity represents the activities the administrator must make to recover the service from the disaster. Referencing to ITIL, this activity can be made using the help of Backup and Restore Activity, and the information in Known Error Database that is included in Problem Management Process.

ER-polling Activity (4) – This activity tests just the single service or instance for reachability after the service has been tried to be recovered. The problem here is to define what kind of wait time is required for the instance data to be transferred to target compute-node machine when re-launching instance. This is because the instance cannot be reachable before the data has been transferred to target machine, and the instance has been booted. As this activity gives results whether the problem has been fixed or not, the alert activity is issued if the result is negative.

Alert Activity (5) – Alert activity invokes the message to system administrator to inform about the current emergency. This activity also can be handled either through the logging activity. To reference to ITIL, this alert goes through the Event management process to trigger the Incident Management.

Logging Activity (6) – All the activities send information to Logging activity. This activity handles the information based on the type of the message received and triggers events based on the received message. Referencing to ITIL, everything that can be measured can be sent to Event Management for tracking and measurement. In a case of errors, the Even Management makes appropriate actions based on the triggering error.

The goal of the second function is to create backups from the instances that are running in the environment. For this function, the sequence chart includes objects called script, nova/backup, move BU and Logging. The “script” object comes from the manual work where the administrator stars the process to create the backups from instances. The nova/backup object is derived from the commands that were used to make a snapshot from the running instance. Moving the backups to a long term storage is called “move BU” object. The objects and source of the objects are represented in figure 20.

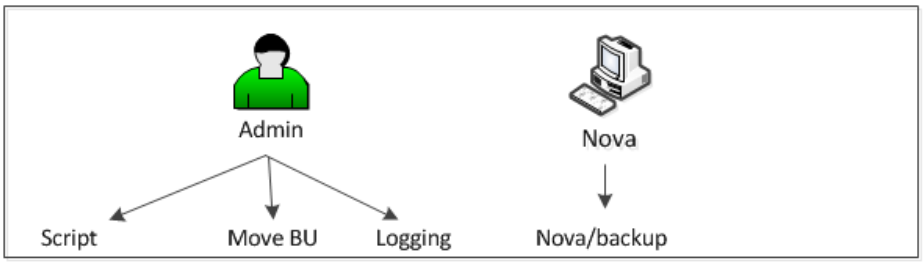

FIGURE 20. Objects for the backup sequence flow.

Figure 21 represents the relationship between the objects for backup script and is visualized in a timeline, as some events or actions between these objects require some event to be finished before the next object can operate.

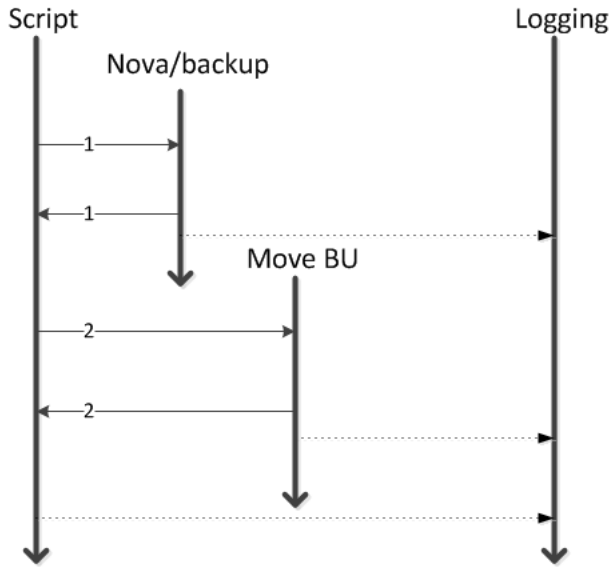

FIGURE 21. Sequence flow for backup script

Figure 21 shows the relationship between objects and these relationships are opened up as follows:

Script & Nova/backup (1) – The controlling script initiates the scheduled backup for instance and/or volume. As the script object can reside in some other location than the nova-api, so the relationship between these two objects require some way to communicate.

Script & Move BU (2) – When the backups from volume and instance are done, the Move BU object moves these fresh backups to the long term storage. The relationship requirement here is the need to be able to move the backups from one location to another and the middle storage might not be in same location as where the script itself is located.

Logging (*) – All of the objects push logging data for the logger object. This data can then be used for error management, measurement and to trigger events. The logger object is responsible for handling the messages that other objects send for it. The relationship between the logger and

other objects require some way for the objects to send the logging data for the logger.

Using the sequence chart that visualizes the relationships between interfaces and knowledge how the instances were manually launched, the activities for the work flow can be made. The result of this will be represented in figure 22 and the activities in the figure are opened up described activity at a time.

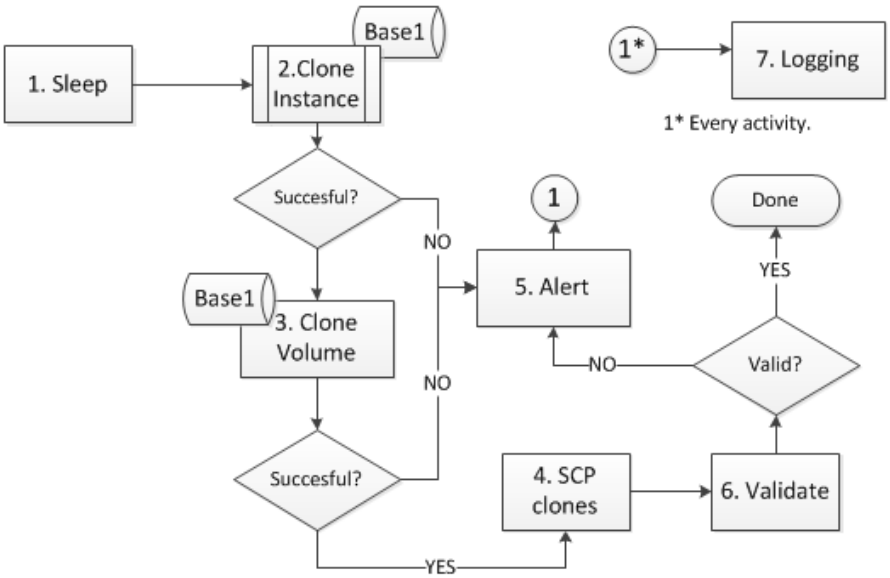

FIGURE 22. Work flow for backup script.

Activities within the backup work flow are described as follows:

Sleep Activity (1) – The sleep activity is something that is required for the script not to trigger the cloning activities too often as this loop would otherwise result in too much consuming servers resources. As the idle time has been passed, the sleep activity triggers the Clone Instance Activity.

Clone Instance Activity (2) – The Clone Instance Activity is responsible for creating the snapshot of the running instance. What instance is to be snapshotted is checked from the database, as there might be different intervals for backup schedule, also the back upping of different instances

can be divided into larger time window to preserve recourses. As the snapshot is done, the information of this newly created image is placed to database in case there were some issues with taking the snapshot from the instance, the alert activity is issued. The Backup and Restore described in ITIL can be used to support this activity.

Clone Volume Activity (3) – Clone volume activity is very much the same as Clone instance activity, except the virtual hard drive of instance is snapshotted.

SCP Clones Activity (4) – The SCP Clones activity is responsible for moving the fresh snapshots to long term storage where this data can be later retrieved if required.

Alert Activity (5) – Alert activity invokes the message to system administrator to inform about the current emergency. This activity also can be handled through the logging activity. The Event Management described in ITIL can be used reference here.

Validate Activity (6) – The Validate activity is there to check the integrity of the backup data, so it will be later usable for disaster recovery or for other purposes. If the data does not pass the validation check, this occurrence results in moving to alert activity.

Logging activity (7) – All the activities send information to Logging activity. This activity handles the information based on the type of the message received and triggers events based on the received message. Referencing to ITIL, everything that can be measured can be sent to Event Management for tracking and measurement. In a case of errors, the Even Management makes appropriate actions based on the triggering error.

Appendix 5. Creating Work Flow (terminating)

In the terminating phase the focus is to create a work flow that describes the steps and activities required to terminate the instance and store the instance data to long term storage in case it is required.

Based on the third scenario and the steps done on manual work, the instance was brought down using just one computer, the nova-api computer. This nova-api machine is described to be a Nova object for the purposes of sequence flow. The same thing is done here as in the running phase, the user, administrator and logging are also made to be objects for the same reasons as described before. The copy object comes from the tasks that are required to be made in different machine than the nova-api machine and is derived from that.

Figure 23 represents the relationship between the objects for terminating script and this is visualized in a timeline, as some events or actions between these objects require some event to be finished before the next object can operate.

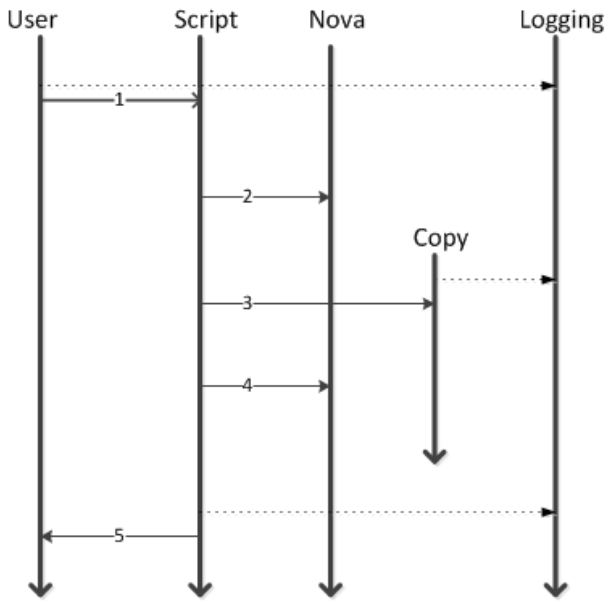

FIGURE 23. Sequence flow for terminating script.

The sequence flow in figure 23 is opened up as follows:

User & Script (1, 5) – User starts the process to stop or terminate the instance using either the web terminal or shell terminal. The relationship between these two objects requires some sort of API solution. After the instance has been stopped or terminated, the script informs the user through the same channel where the request came.

Script & Nova (2, 4) – The script uses the nova object to stop the requested instance. Most likely the script and the nova-api are located in different locations so the relationship between these two objects requires some sort of API solution. Later as the instance and volume data has been copied and moved to long term storage, the script issues the instance and volume to be terminated.

Script & Copy (3) – If the instance and volume data were to be saved to long term storage, the instance and volume data will be first snapshotted and transferred to long term storage. The relationship between these two objects requires some sort of an API solution.

Logging – All of the objects push logging data for the logger object. This data can then be used for error management, measurement and to trigger events. The logger object is responsible for handling the messages that other objects send for it. The relationship between the logger and other objects require some way for the objects to send the logging data for the logger.

Using the sequence chart that visualizes the relationships between interfaces and knowledge how the instances were manually launched, the activities for the work flow can be made. The result of this will be represented in figure 24. Work flow for terminating script and the activities in the figure are opened up described activity at a time.

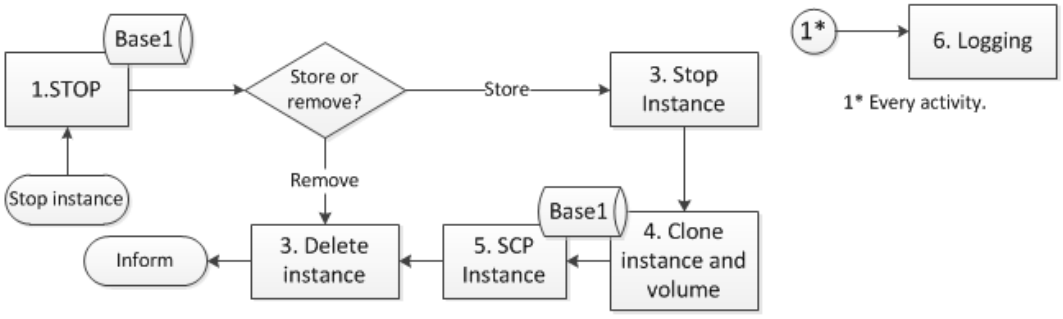

FIGURE 24. Work flow for terminating script.

Activities within the terminating work flow are described as follows:

Stop Activity (1) – The Stop activity initiates the whole process to terminate the instance. It listens to incoming messages that states whether to stop or terminate the instance, and what instance. When this activity receives trigger to stop the instance, it cross- references the information in database to make the decision whether to make a backup from the instance before termination or not. Alternatively the instance could be just stopped without removing the instance data. When referencing to ITIL, this can represent the situation where the Service Desk has gotten the request to stop the service and the request will be forwarded to Server Management that will handle the process to terminate the service at the system.

Delete Instance Activity (2) – The Delete Instance activity either deletes the instance related data from computing environment or just shuts down the instance based on the information on the trigger that activated this activity.

Stop Instance Activity (3) – The Stop Instance shuts down the instance before moving to the next activity.

Clone instance and Volume Activity (4) – Clone instance and Volume activities function in the same way as in the activities in the backup phase. The Backup and Restore activity can be used to support this activity.

SCP Instance Activity (5) – SCP instance activity moves the instance related data to long term storage and functions in the same way as the activities in the backup phase.

Logging Activity (6) – All the activities send information to Logging activity. This activity handles the information based on the type of the message received and triggers events based on the received message. Referencing to ITIL, everything that can be measured can be sent to Event Management for tracking and measurement. In a case of errors, the Even Management makes appropriate actions based on the triggering error.

Appendix 6. Tools for Work flow (recovery)

The previously represented figure 19. Instances disaster recovery work flow can be found in Appendix 4. Creating Work Flow (Running) and is used as a reference point to find the required tools to accomplish the described activity. The activities are opened up activity at a time and the tools to accomplish the functions in activities are represented.

The sleep activity (1) is responsible for timing the backup schedule. To make this activity work, there is a requirement for some way to schedule the initiation of polling activity. The scheduling could be easily done using the server's cron job to schedule the initiation of the script, but there must be also a way to accomplish this using the Python. Python equivalent solution is to use python time module, with which the scheduling of the polling activity can be made (Python Software Foundation, n.d). Figure 25 summarizes the described solutions for this activity.

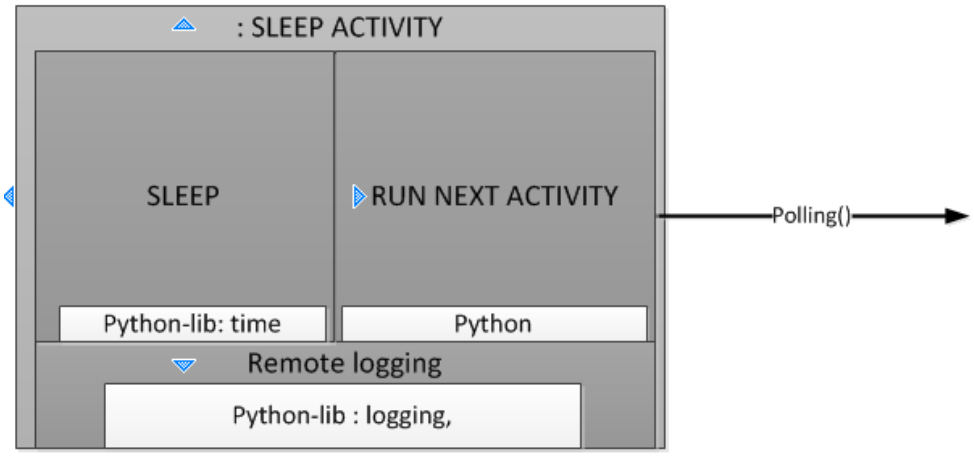

FIGURE 25. Proposed solution for activity 1. in the recovery script

The polling activity (2) is used to check the reachability of services and instances, but these checks can be somewhat tricky to achieve in a way that it does not give false positive results. The requirement to make this activity to work is to find a way how the tools installed to the server can be used and with which commands can be issued to remote locations. The solution is to use the python "subprocess" module with which the commands can be issued for local

machine (Python Software Foundation, n.d). The commands to remote locations can be done using Fabric, which is already described before. When using the tools either in local or remote machine, the output of these tools needs to be parsed to determine whether the instance or service is reachable. For example, in a case if the instance is not responding, all the services used between the script and instance also need to be verified to minimize the false positives resulted from faulty service in between. What is important is to pin point the source of the problem thus actions to resolve it can be made properly. The activity also checks the database what services are scheduled to be checked, and with this the different polling intervals for services can be made to minimize the timely spam traffic made by the reachability checks. The database connection can be made using the python sqlite3 module. Figure 26 summarizes the described solutions for this activity.

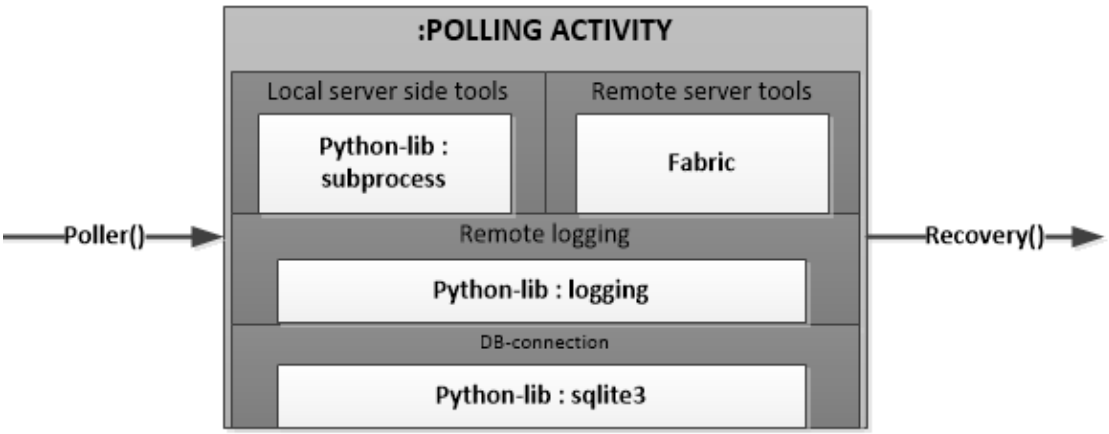

FIGURE 26. Proposed solution for activity 2. in the recovery script

In a case the polling activity finds one component to be unreachable, the recovery activity (3) tries to resolve this issue. The recovery mechanism decides the actions based on the occurred failure; the failures can be divided into two different categories. In scenario where the infrastructure is healthy, but the instance is unreachable, the instance can be first hard rebooted using fabric or re-launched using the latest available backup based on information in database if the reboot did not provide the wanted results. In a case where one or more components seem to have failed at the infrastructure, the recovery

activity will try to restart the services at servers where the service failed to run. The restarting of services can be achieved using the commands with Fabric.

ER-polling activity (4) is the same as the polling activity with the exception that the failed service or instance will be polled. In a case the service or instance is still unreachable, the alert activity will be used. Figure 27 summarizes the described solutions for this activity.

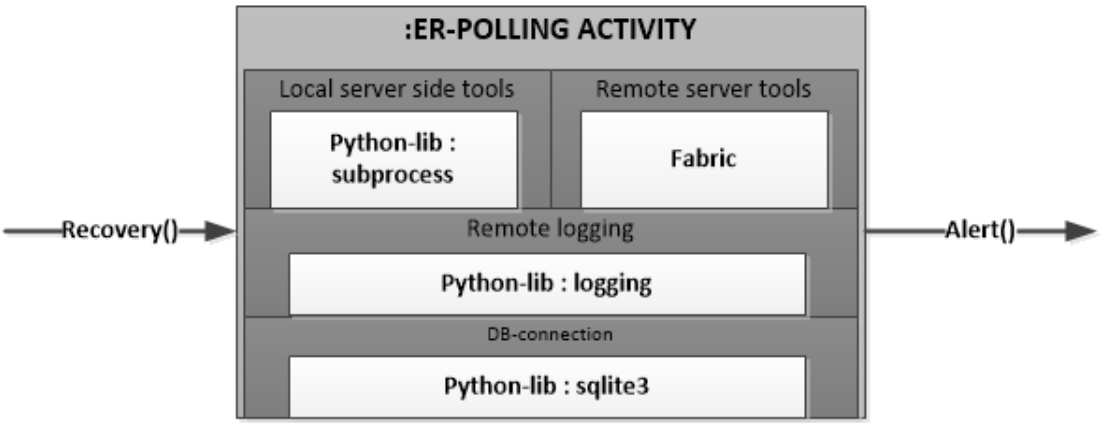

FIGURE 27. Proposed solution for activity 4. in the recovery script

Alert activity (5) is there to send the error message to system administrator. The problem here is to find a way to send the message to administrators in a way that the issue will be noticed as fast as possible. To ensure that the information receives the administrators, multiple communication solutions must be used. The first way is to use the logging activity to forward the message. The second is to send the message through mail using python smtplib module (Python Software Foundation, n.d). Additional ways to send the message are to send it through system that uses SMS to deliver the messages; this is in a case the network is unable to forward the messages through physical media. The SMS solution is just represented as a possibility to trigger this event, the whole SMS process and system would require its own activities and systems to be solved and possibly could even be large system enough to cover own thesis. Figure 28 summarizes the described solutions for this activity.

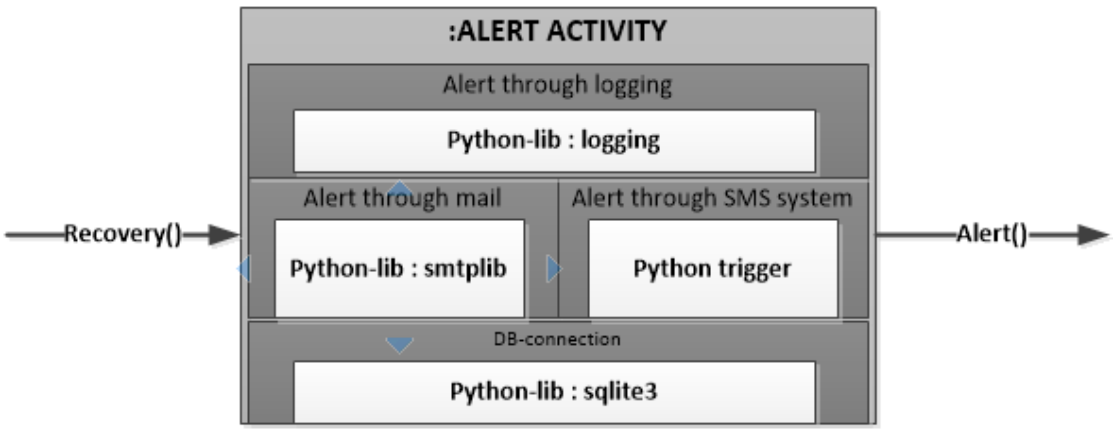

FIGURE 28. Proposed solution for activity 5. in the recovery script

The logging activity is opened up in in Appendix 9.

Appendix 7. Tools for the Work Flow (backup)

The previously represented figure 22. Work flow for backup script can be found in Appendix 4. Creating Work Flow (Running) and is used as a reminder to previously presented work flow that and used as a reference point to find the required tools to accomplish the described activity. The activities are opened up activity at a time and the tools to accomplish the functions in activities are represented.

For the backup script, the sleep activity (1) handles the timing of backups. The activity faces the same problems and solutions represented in the work flow on the recovery script and the solution is to use the python time module. Figure 29 summarizes the described solutions for this activity.

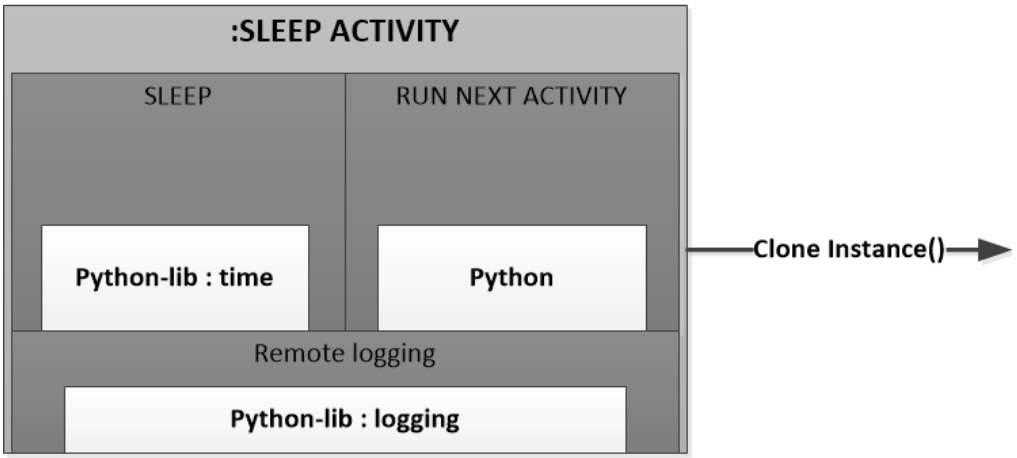

FIGURE 29. Proposed solution for activity 1. in the backup script

Clone Instance activity (2) represents the actions required to take the snapshot from the running instance. The problem here is to take a live backup from the instance that is running in the environment. The problem can be solved issuing the commands for the nova-api machine using fabric, and the activity checks the database to decide what instances are required to be snapshotted. Figure 30 summarizes the described solutions for this activity.

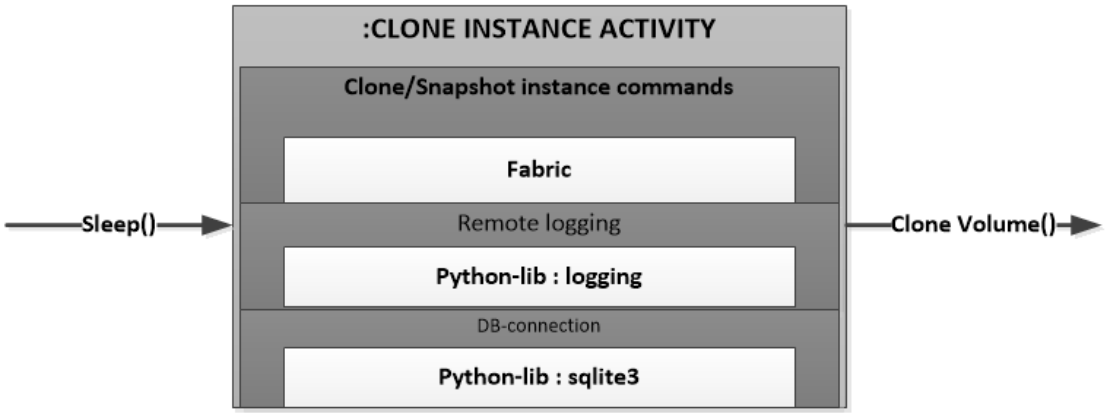

FIGURE 30. Proposed solution for activity 2. in the backup script

The clone volume activity (3) handles the tasks to create backup from the volume designated for the instance and it works very much the same way as the clone instance activity (2), except that it creates the backup from the volume designated for the instance. The backups can be made using the same tools as represented in previous activity. Figure 31 summarizes the described solutions for this activity.

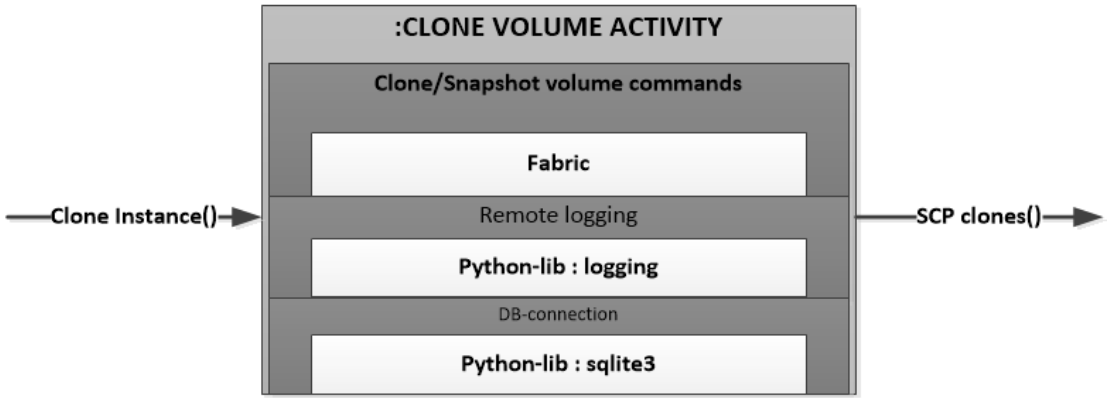

FIGURE 31. Proposed solution for activity 3. in the backup script

The SCP clones activity (4) is used to move the backup data to long term storage. The activity can be made to work using the fabric that is used to handle the commands in remote locations to move the data using SCP, secure copy. Figure 32 summarizes the described solutions for this activity.

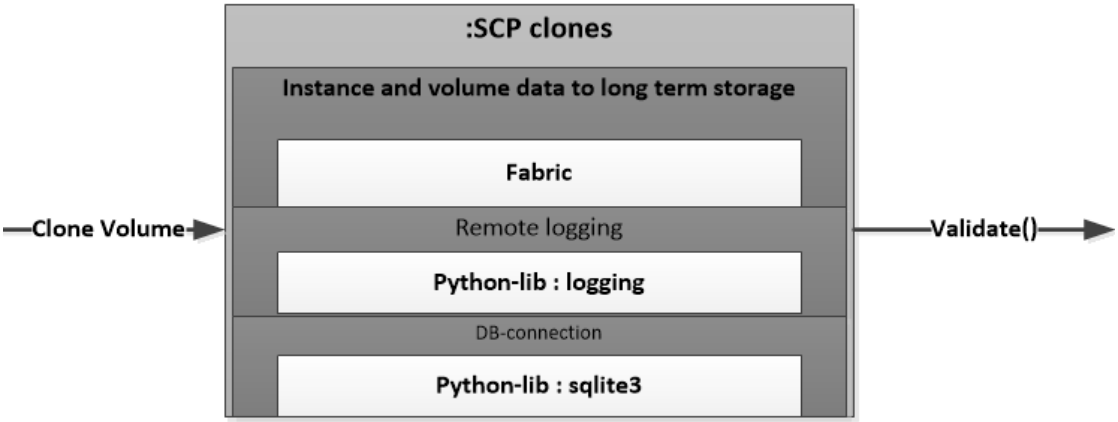

FIGURE 32. Proposed solution for activity 4. in the backup script

The Alert activity (5) is there to send the error message to system administrator. The problem here is to find a way to send the message to administrators in a way that the issue will be noticed as fast as possible. To ensure that the information receives the administrators, multiple communication solutions must be used. The first way is to use the logging activity to forward the message. The second is to send the message through mail using python smtplib module (Python Software Foundation, n.d). Additional ways to send the message are to send it through system that uses SMS to deliver the messages; this is in the case the network is unable to forward the messages through physical media. The SMS solution is just represented as an idea. Figure 33 summarizes the described solutions for this activity.

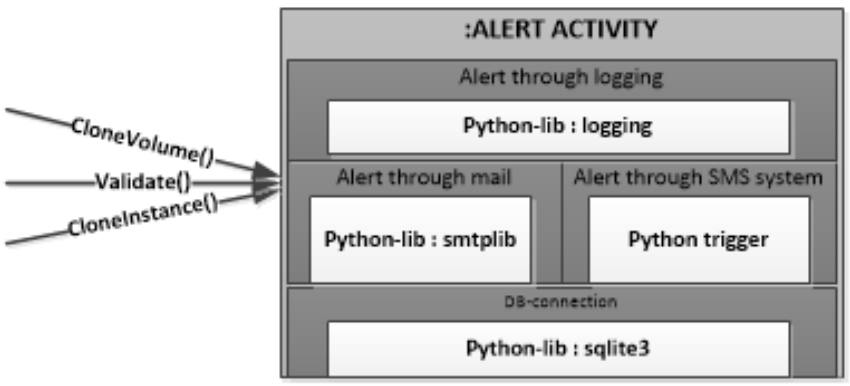

FIGURE 33. Proposed solution for activity 5. in the backup script

Validation activity (6) handles the verification of the data that has been transferred to long term storage. The problem here is to find a way to verify the integrity of BLOB-object. The first step of verifying the data is to check the size of the file; if there is a difference in size, the file has been altered. Even if the size has not been changed, the data can still be changed somehow, thus there is also a need to make and compare the file's hash before and after. The hash from the file will be created before the file has been transferred and will be recalculated after the file transfer using the Python zlib library (Python Software Foundation, n.d). Figure 34 summarizes the described solutions for this activity.

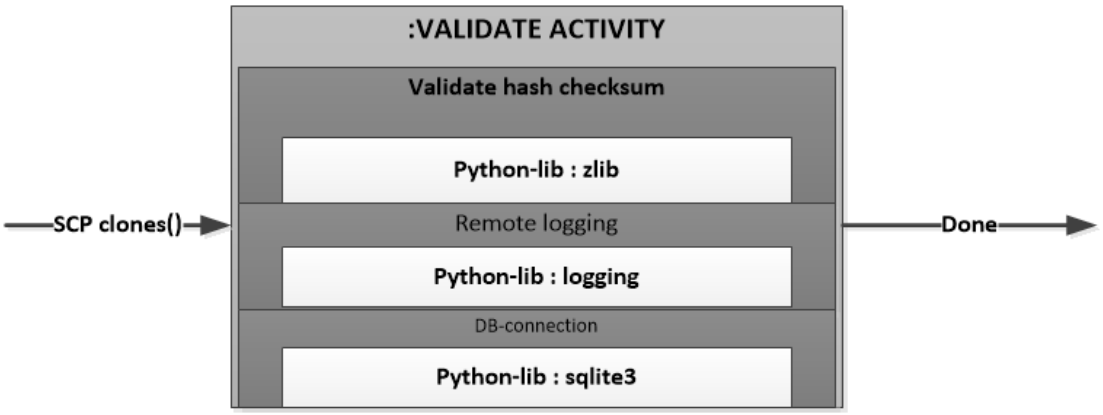

FIGURE 34. Proposed solution for activity 6. in the backup script

Appendix 8. Tools for the Work flow (terminating)

The previously represented figure 24. Work flow for terminating script can be found in Appendix 5. Creating Work Flow (terminating) and is used as a reminder to the previously presented work flow that is used as a reference point to find the required tools to accomplish the described activity. The activities are opened up activity at a time and the tools to accomplish the functions in activities are represented.

The stop activity (1) gets the trigger to stop the instance in same way as the first activity in startup script. The stop activity listen incoming triggers that are originated from the web service and it makes decision whether to save or delete the instance data based on the information on trigger. The activity can be made using the same solutions as described in the first activity in chapter 5.5.2. Figure 35 summarizes the described solutions for this activity.

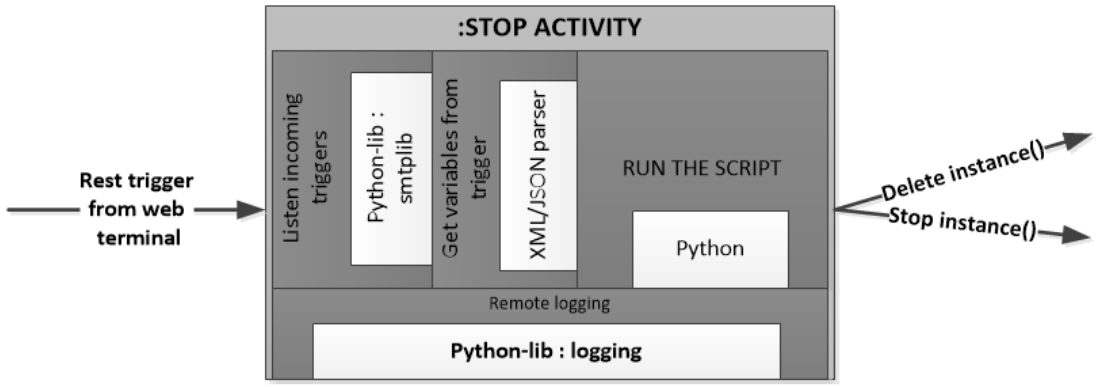

FIGURE 35. Proposed solution for activity 1. in the terminating script

Stop Instance (2) activity stops the instance if the instance were requested to be suspended, not terminated. The instance can be suspended using fabric to issue the commands to nova-api machine in the same way as described in the running script. Figure 36 summarizes the described solutions for this activity.

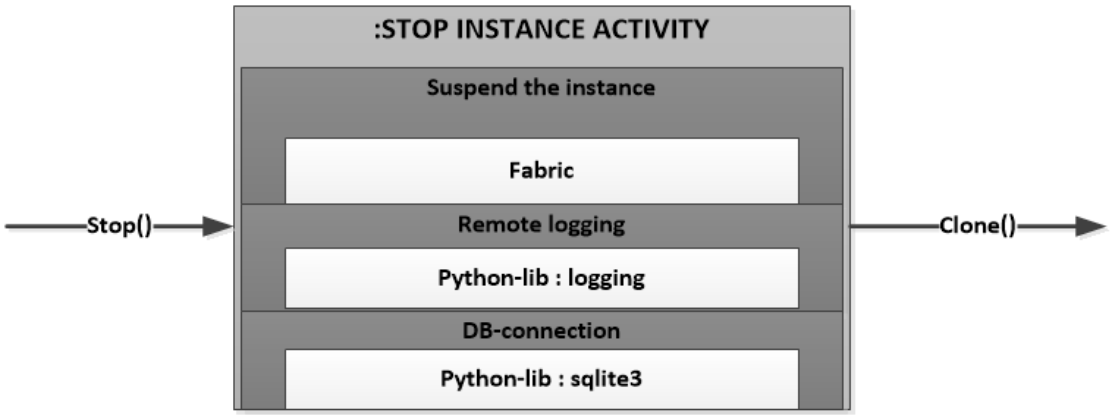

FIGURE 36. Proposed solution for activity 2. in the terminating script

Clone instance and volume (4) activity handles the tasks to create the last backup from the instance and from the volume related to instance. This can be made using fabric as described in the backup script. Figure 37 summarizes the described solutions for this activity.

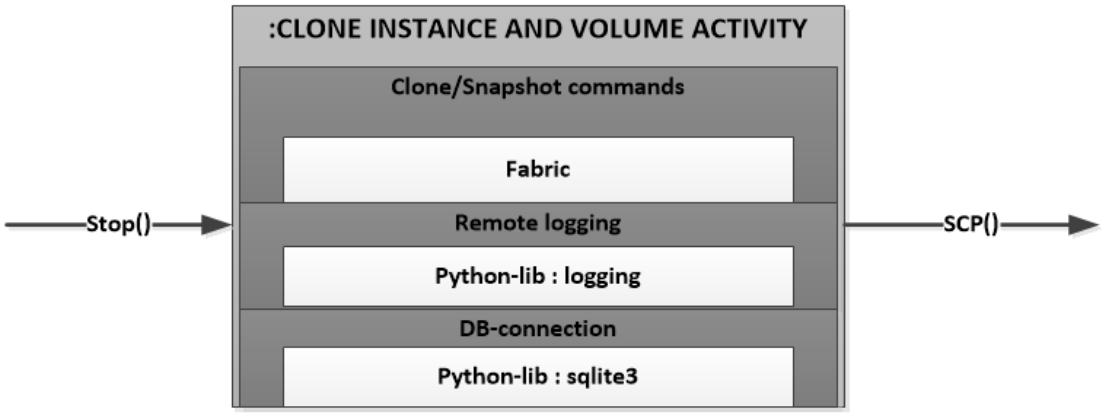

FIGURE 37. Proposed solution for activity 4. in the terminating script

SCP instance (5) activity handles the tasks to transfer the instance to long term storage. This can made using the fabric as described in SCP clones activity in backup script. Figure 38 summarizes the described solutions for this activity.

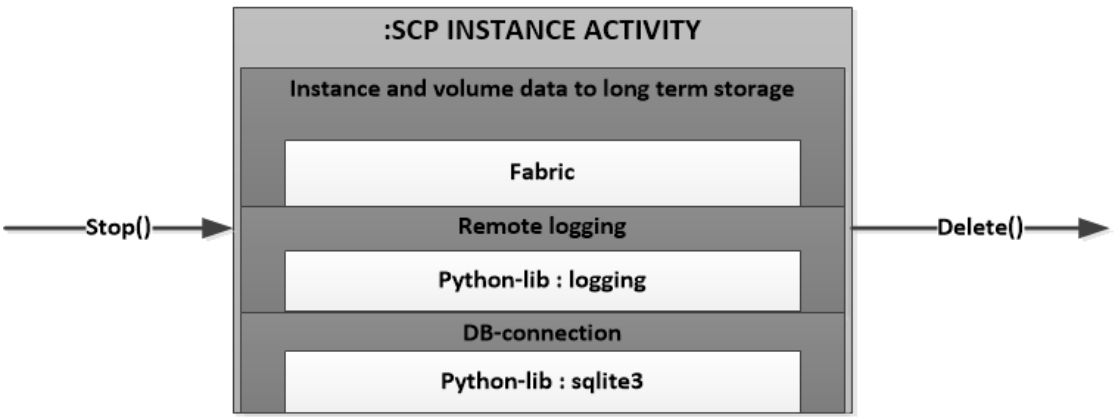

FIGURE 38. Proposed solution for activity 5. in the terminating script

Delete instance activity (3) handles the instance removal from the system if it were requested to be removed. The removal of instance can be made using the fabric to issue the instance terminating commands to nova-api machine. Figure 39 summarizes the described solutions for this activity.

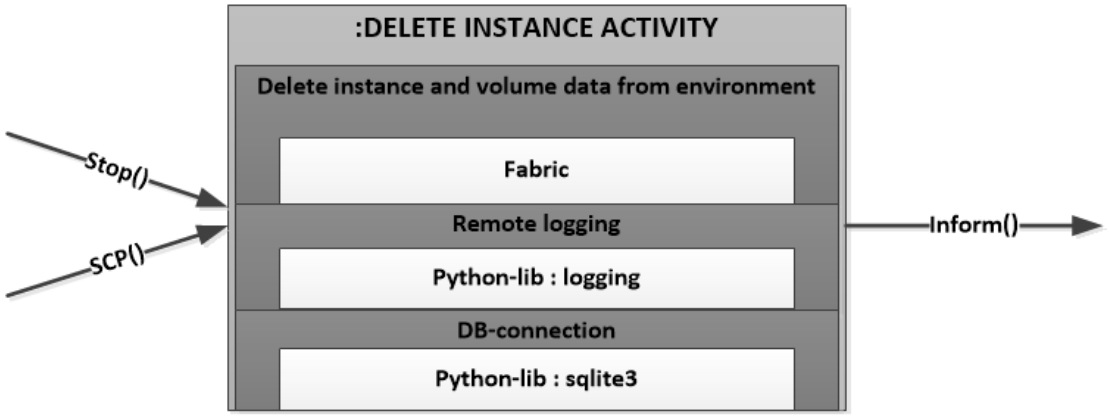

FIGURE 39. Proposed solution for activity 3. in the terminating script

As the instance has been terminated or suspended, the user will be informed using the same channels as the ones the request came through. This activity is done same way as described in in startup phase; placing the information to database where the web terminal can get this information. Figure 40 summarizes the described solutions for this activity.

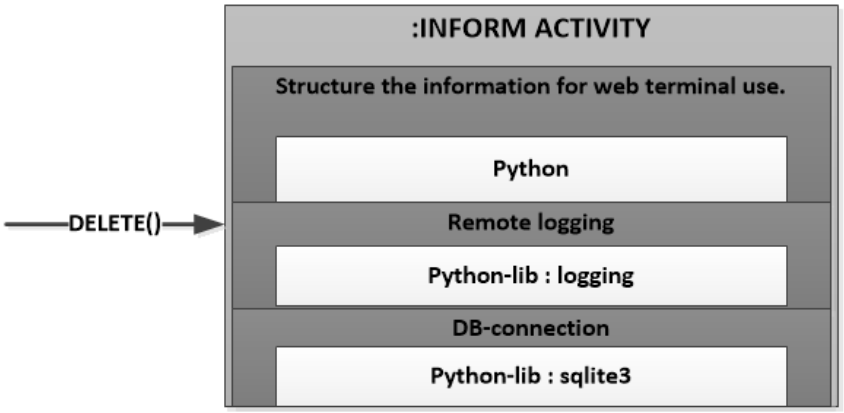

FIGURE 40. Proposed solution for inform activity in the terminating script

Appendix 9. Tools for the Work flow (logging)

The logging activity can be found on all the three different phases of an instance's life and this activity will be opened up here in more detailed terms. There are two possible outcomes what this logging activity can be. The simple version is where this logging activity is just used as a centralized location where the logs are dumped and other services use this information. The complicated way is where this activity is used to manage the logging information and make actions based on this information. In the complicated way, the logging activity can be used as event management to trigger processes/events based on the logging information. The second solution follows the ITIL event management process.

The requirement for the simple logging activity is that it must be able to receive the information that other activities send for this activity, it must parse the information to human readable format when saving to log file and be able to categorize the incoming information. The solution is to use the python library logging, with which the centralized logging can be achieved and the sqlite3 library can be used to support the activity. Figure 41 represents the summary of solutions for the logging activity.

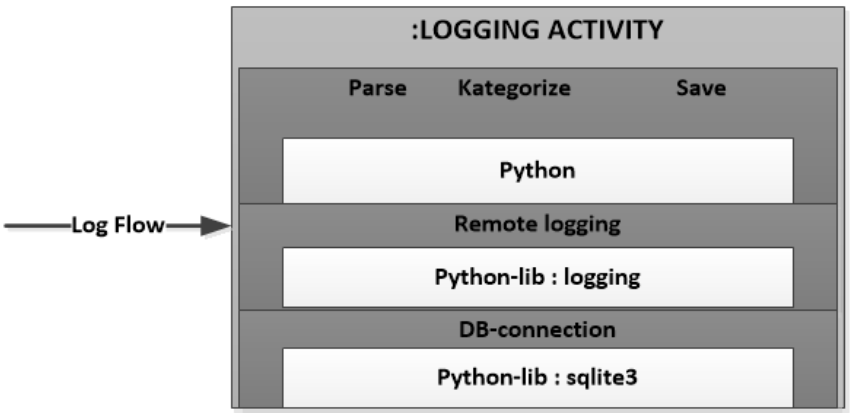

FIGURE 41. Proposed solution for Logging activity