Hochschule SRH Heidelberg

Internship Report

Access Load Balancer via REST calls and Automation of Network Diagrams

Author:

Alpana Chaphalkar

(alpana.chaphalkar@sap.com)

Supervisor:

Ionut Roibu

(ionut.roibu@sap.com)

Internship at

SAP Hybris

From July 2016 to September 2016

A report submitted in fulfilment of the requirements

for the Internship of Master of Science

in the

Informatics Department at

SRH Hochschule Heidelberg

September 30, 2016

DECLARATION OF AUTHORSHIP AND CONFIDENTIALITY

I, ALPANA CHAPHALKAR, declare that the content presented in this Internship Report titled, "Access Load Balancer via REST API and Automation of Network Diagrams" is a work of my own. I hereby confirm that:

- The entire work presented in this report was done during the employment at SAP Hybris.
- The work of others has been clearly quoted and source of information is also provided wherever required. Apart from these quotations, all the specified work is entirely done by me. All main sources of help have been acknowledged.
- The data used in this report is not customer data of SAP Hybris. Only internal project data and documentation of SAP Hybris is used and appropriate references for this data is provided. I hereby confirm that I have not used any customer confidential data of SAP Hybris in this report.

Signature:

____

Date:

____

ABSTRACT

Access Load Balancer via REST calls and

Automation of Network Diagrams

BY ALPANA CHAPHALKAR

SAP Hybris is known for its enterprise multi-channel e-commerce and product content management software product [1]. Platform architecture and automation (DevOps) team in the hybris cloud services (hCS) department of SAP Hybris is responsible for implementing network architecture and application environments of e-commerce websites based on B2C (business to consumer) and B2B (business to business) customer requirements. This team also does deploying of product content once the e-commerce website is implemented and supporting the implemented and deployed e-commerce shop. The tasks presented in this internship report comes under automation section of hCS DevOps team. In the era of agile software development, continuous delivery of software product based on customer's constantly changing requirements is a challenge for many enterprises and this challenge is resolved to major extent by DevOps paradigm. DevOps paradigm aims for fast and frequent releases of software product by developers in collaboration with operations people. Traditionally, setup of development and operations is split across various departments which makes much harder and difficult to efficiently satisfy the constantly changing and growing requirements of customers. Such kind of situations are dealt using DevOps team, because in DevOps development and operations tasks goes hand in hand. In order to achieve DevOps objective, automation of repeatable manual processes are carried out which are error-prone, slow and costly [2]. DevOps is a theoretical paradigm to facilitate continuous delivery of the product while automation is one of the technique to fulfill DevOps objective. In order to automate the manual processes, REST API web services are used. In this report chapter 1 describes the overview of hybris platform architecture required for the SAP Hybris e-commerce product. Chapter 2 describes the implementation of REST calls to access the information from load balancer in detail. Chapter 3 describes the logic of automated network diagrams. Chapter 4 provides the summary of presented internship tasks and at last references are given.

Table of Contents

List of Figures

List of Tables

Chapter 1

Delivery of Hybris Platform

1.1. Introduction

This chapter provides an overview of setup of hybris platform required for the delivery of e-commerce product. The setup of hybris platform for a customer is done by multiple teams of hybris cloud services collaboratively. The processes performed for implementing hybris platform mentioned here are already automated while some are done manually. The figures or diagrams provided in this chapter is the copyright information of SAP Hybris and the references for these are mentioned in the references chapter of this report.

1.2. Hybris Platform

The complete SAP Hybris commerce suite is built upon hybris platform. Following figure [3] shows major parts in hybris e-commerce suite:

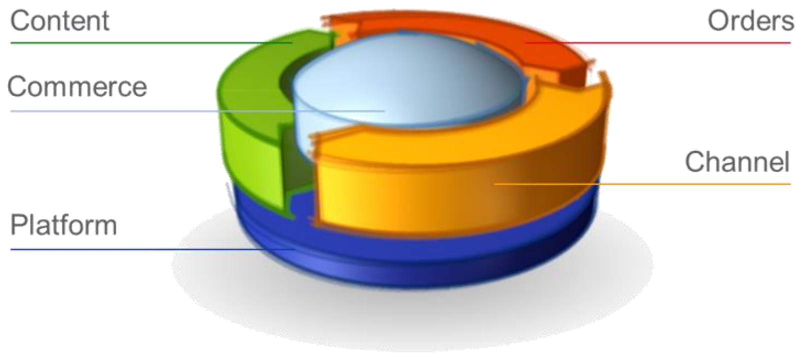


Figure 1.2-1 Hybris Cake

Platform architecture and automation (DevOps) team is responsible for the delivery of hybris platform of customers.

1.3. Platform Architecture Overview

The platform is implemented after a project sign-off with customer. There can be multiple projects with a single customer. Following sections [4] provide an overview of customer projects and their system landscape:

1.3.1. Datacenters

All customer projects are hosted on following datacenters [4] of SAP Hybris:

- Boston (MA)
- Frankfurt (FR)

- SAP HEC
- Legacy Systems

1.3.2. Customer Environments

Every customer project consists of following environments [5], [6]:

Development	QA	Staging	Production
Development environment is mostly used by customer for creating, modifying, testing and maintenance of software.	QA environment is optional and is used for functional, performance testing or quality assurance.	Staging environment is exact replica of production environment.	Production environment is used by end-users and end users can be customer or customers of customer.

Table 1.3-1 Customer Project Environments

Generally, a single customer project of multiple environments is hosted on a single datacenter.

1.3.3. Hosting models for customer or partner access

To provide the flexibility to partners or customers to decide on their level of access of control, hybris has provided these hosting models. Based on these, customer or partner can have access to certain project environment systems and can do deployment of their code on accessible systems. There are 2 hosting models [7] which are described as follows:

Closed Model	Open Model
• Customer/Partner Shell Access: Development • Staging, Production Deployments by hybris	• Customer/Partner Shell Access: Dev, Staging & Production • Initial Demo Installation hybris

Table 1.3-2 Hybris Hosting Models

1.3.4. Architecture

Hybris platform architecture is a 3 tier architecture. Every customer environment follows this 3 tier architecture and every layer of architecture is a cluster of servers. Following figure [6] shows the different layers of architecture:

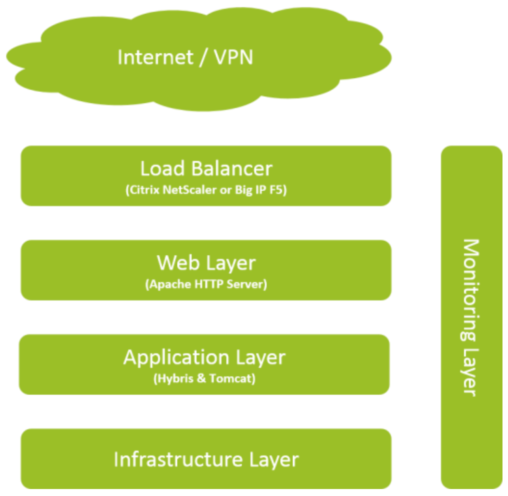


Figure 1.3-1 Hybris Architectural Layers

1.3.5. Hybris Platform Architecture Traffic Flow

The traffic flow from one layer to another through cluster of servers is shown in following figure:

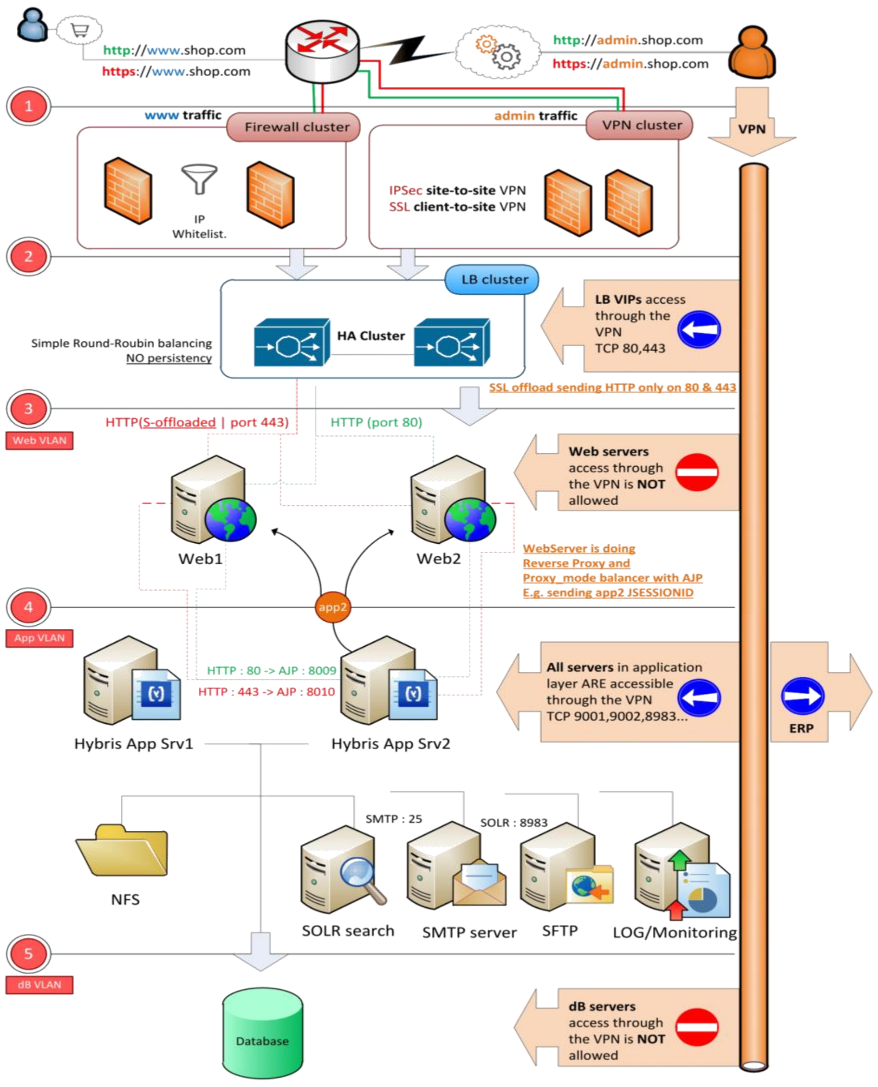


Figure 1.3-2 Hybris Traffic Flow

Following sections provide the overview of main components and their respective roles in traffic flow.

1.3.5.1. Overview of load balancer

Load balancer is a hardware which is responsible for load balancing the traffic from virtual servers to the web server and application server pool [8]. It handles the session stickiness but only on web servers. In the chapter 2 of this report, detailed information on load balancer has been given.

1.3.5.2. Server types

Following table describes the different servers [4] and their role in the traffic flow of hybris platform architecture:

Server type	Purpose or role
Application servers or admin servers	• Serve both the frontend and backend site. • AJP connector is used for load balancing.
Web servers	• Proxies all inbound requests to application or admin servers. • Load balancing traffic from the load balancer to the application server pool. • Redirects and rewrites for http/s requests.
Solr servers or search servers	• Retains search results for the hybris application. • Powers the search and navigation features on the website.
Datahub servers	• Intermediate platform.
Database servers	• Enables integration between hybris and customer's SAP ERP systems (not hosted by hybris cloud services).
SMTP servers	• Acts as a mail server.
Dynatrace Servers	• Used for performance monitoring. • Performs data analysis.
Dynatrace collectors	• Collects data and sends to the dynatrace server.

Table 1.3-3 Server types

1.4. Responsible Teams

As aforementioned, multiple teams at hybris cloud services department of SAP Hybris are responsible for delivering the hybris platform and every team has different set of responsibilities. Although responsibilities of teams are different, teams work in cooperation with each other to deliver the best quality hybris platform. Following are the teams and their responsibilities for delivering the hybris platform:

Team	Responsibilities
Hybris System Administration (hCS Sys Admin)	Configuring the systems required for the hybris platform like web servers, application servers, load balancers, dynatrace servers, etc. Providing access to these systems [9].
Platform Implementation Engineering (hCS PIE)	Installing and deploying hybris application, deploying customer specific applications for customers [10].
Platform Service Engineering (hCS PSE)	Managing hybris platforms for hybris customers as a service with highest quality and support [11].
Platform Quality Engineering (hCS PQE)	Developing tools and services to improve quality of hybris platform and detecting problems in the platform before customer does [12].
Platform Monitoring Engineering (hCS PME)	Monitoring tools and softwares like OpsView, dynaTrace, Splunk. These tools are used for monitoring the platform server's disk usage, CPU performance, etc [13].

Table 1.4-1 Hybris cloud services teams

1.5. Summary

The next chapter is on the REST API for accessing the load balancer and in this chapter, the purpose of load balancer and where it is used in the hybris platform architecture has been stated. Chapter 3 on network diagrams requires basic understanding of hybris platform architecture. In a nutshell, this chapter gives an overview of complete hybris platform architecture such that the terms and concepts pertaining to it described in the following chapters can be comprehended easily.

Chapter 2

Access to Load Balancer via REST calls

2.1. Problem Statement

As mentioned in the previous chapter, the purpose of load balancer is to balance the traffic load from virtual IP to web server pool. Responsible team for configuring load balancer and servers used in hybris platform architecture is hCS Sys Admin while installing hybris application components on the configured servers is responsibility of hCS PIE team. As part of hCS DevOps department, hCS PIE team has the resources to automate the manual processes of installing the hybris application. Following are the existing automated processes:

- Validation checks for configured servers.
- Installing components required for deploying hybris application on configured servers.
- Deploying customer specific application components on configured servers and making sure the shop websites are delivered successfully as requested by customers.

During the deployment and installation of hybris application components, there is often possibility of server status down or the configured service type of a server is not configured as per requirement which causes failure in load balancing the traffic on different web servers or application servers. In such cases, hCS sys admin team has to check the load balancer configurations which is a manual process and causes the delay in the installation and deployment of hybris platform for customers. Moreover, it was difficult to figure out at point whether load balancer configurations are incorrect or application components configurations are incorrect because hCS PIE team did not have the access of information from load balancer and in case of any problems they had to disturb hCS sys admin team.

Hence in order to avoid time consuming manual process of checking and figuring out the problem cause, access to load balancer configuration information was significant. To access this information from load balancer, REST API is used and to automate this process nightly cron jobs are used.

This section clearly explains the motivation behind the access to load balancer via REST API. Further sections in this chapter provide the technical and logical details about the load balancer and its access via REST API.

2.2. Load Balancer (LB)

The purpose of using the load balancer is to improve the performance and security of applications by managing the distribution of network or data traffic and services [14]. Load balancer is the essence of increasing the capacity (number of users accessing applications at the same time) and reliability of applications [15]. In SAP Hybris for hybris e-commerce application, based on customer requirements either citrix's application delivery controller (ADC) product (load balancer) named NetScaler or f5 network's ADC product (load balancer) named Big-IP is used. For detailed mechanism of both load balancers, references of citrix's load balancer NetScaler and f5 network's load balancer Big-IP are given.

2.3. Logic of REST calls

Both NetScaler and Big-IP load balancers are designed to support the REST web services. Citrix has provided nitro [16] as a REST API for accessing the NetScaler load balancer while f5 has provided iControl [17] as a REST API for accessing the Big-IP load balancer. These REST API libraries must be included in SAP Hybris legacy testing tool called yTestRunner as the test scripts used by this tool are also used by cron jobs to publish the testing results on the confluence (wiki.hybris.com) on a daily basis every night. Hybris application is based on java, the maven dependencies of both load balancer's REST API for java was creating the conflict with the dependencies of legacy testing tool yTestRunner. In order to detect if there are any problems related to load balancing while installing or deploying application components, one piece of code for accessing both load balancers is implemented and this piece of code called as REST API wrapper which is basically making the rest calls to both load balancer's restful architecture using existing maven dependencies of yTestRunner.

The cron job execution script is implemented by PQE team and as part of the internship task the implemented REST API wrapper code is of my own. For implementing this code org.apache library [18] of java is used. Backend logic is implemented using java and for frontend the wiki markup is produced in order to publish the fetched data on confluence. Basically backend java logic returns the data in the wiki markup format using string class of java. For maintaining all the test scripts used by yTestRunner where this REST API wrapper code resides, enterprise-stash repository system is used and git as a version control system is used. For developing these test scripts and REST API wrapper, netbeans integrated development environment is used. Following is the URL of enterprise-stash repository to REST API wrapper code:

https://enterprise-stash.hybris.com/projects/HCS-PDO-QE/repos/ytestrunner/browse/src/main/java/com/hybris/ms/qa/testrunner/genericsteps/loadbalancer/restapiwrapper

Following flowchart describes the logic used for implementing the API:

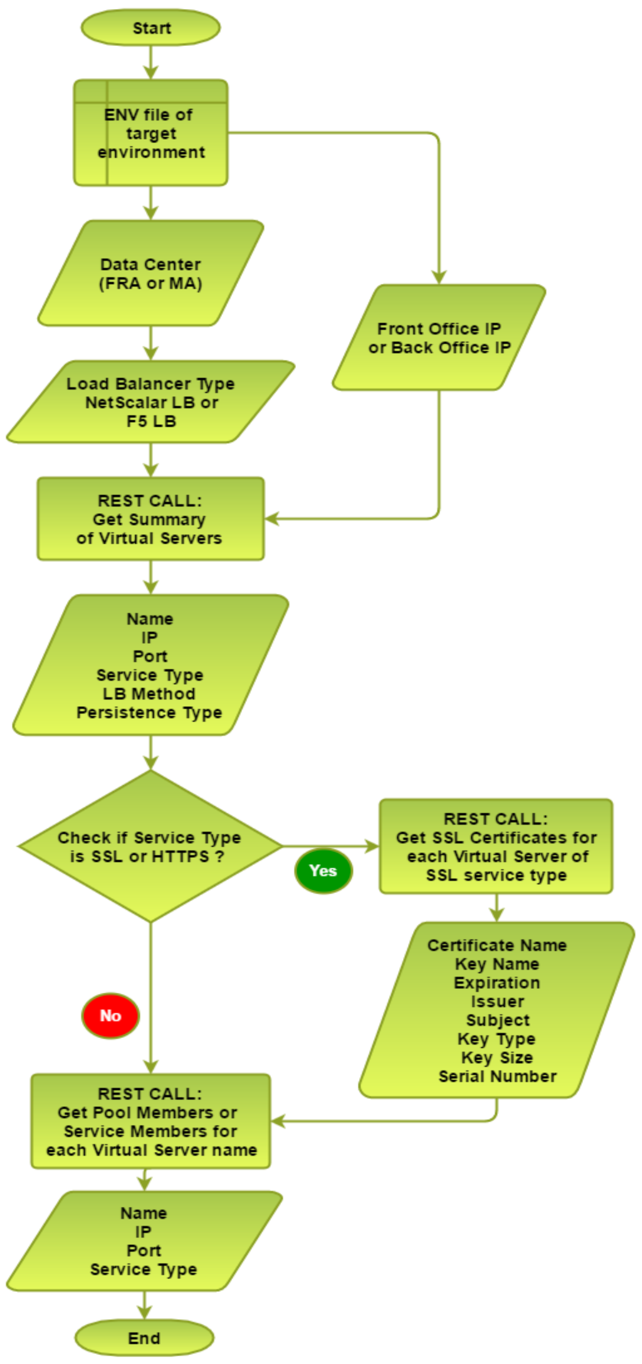

Start
ENV file of target environment
Data Center (FRA or MA)
Front Office IP or Back Office IP
Load Balancer Type NetScalar LB or F5 LB
REST CALL: Get Summary of Virtual Servers
Name
IP
Port
Service Type
LB Method
Persistence Type
Check if Service Type is SSL or HTTPS ?
Yes
REST CALL: Get SSL Certificates for each Virtual Server of SSL service type
Certificate Name
Key Name
Expiration
Issuer
Subject
Key Type
Key Size
Serial Number
No
REST CALL: Get Pool Members or Service Members for each Virtual Server name
Name
IP
Port
Service Type
End

Figure 2.3-1 REST API wrapper flowchart

In above flowchart, target environment is the environment of customer project and the information of each environment of customer project is stored in environment specification file. All the environment specifications files of customers are stored on central repository. From the target environment, following is the required information for accessing the load balancer [19]:

- Internal IP address of front office domain or URL of shop.
- Internal IP address of back office domain or URL of shop.
- Datacenter where customer environments are hosted.

Based on the datacenter information, load balancer type is decided i.e. whether it's a NetScaler LB or Big-IP LB. Following table describe the java classes used and implemented:

Java Classes and their descriptions:

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.CreateWikiMarkup.java**
  This java class is implemented to create wiki markup after getting the data from LB.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.LoadBalancerType.java**
  This java class is implemented to decide the load balancer type based on the data center, front office, and back office IP information.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.RestApiCommonFunctionalities.java**
  This java class is implemented to create self-signed certificates for https requests of REST calls and also some common functionalities which are required by NetScaler and F5 classes

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.NetScalarLbGetSummaryInfoOfVirtualServers.java**
  This java class is implemented to get the list of virtual servers and their summary information from NetScaler LB based on front office or back office IP of target environment.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.NetScalarLbGetServiceGroupMembersOfVirtualServers.java**
  This java class is implemented to get the list of service pool members (real servers) from NetScaler LB based on the list of virtual servers for a target environment.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.NetScalarLbGetCertificateInfoOfSslVirtualServers.java**
  This java class is implemented to get the list of certificates and their details from NetScaler LB based on the list of virtual servers of service type SSL or HTTPS for a target environment.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.F5LbGetSummaryInfoOfVirtualServers.java**
  This java class is implemented to get the list of virtual servers and their summary information from Big-IP LB based on front office or back office IP of target environment.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.F5LbGetServiceGroupMembersOfVirtualServers.java**
  This java class is implemented to get the list of service pool members (real servers) from Big-IP LB based on the list of virtual servers for a target environment.

- **com.hybris.ms.qa.testrunner.genericsteps.loadbalancer.restapiwrapper.F5LbGetCertificateInfoOfSslVirtualServers.java**

This java class is implemented to get the list of certificates and their details from Big-IP LB based on the list of virtual servers of service type SSL or HTTPS for a target environment.

- **com.hybris.ms.qa.testrunner.environment.TargetEnvironment.java**
  This class is used to get the target environment ID of customer project.

- **com.hybris.ms.qa.testrunner.environment.TargetEnvironmentTools.java**
  This class is used to get the front office and back office IP of customer project based on the target environment ID of customer project.

- **com.hybris.ms.qa.forms.envdesc.EnvironmentDescription.java**
  This class is used to publish the load balancer configuration information in wiki markup format (CreateWikiMarkup.java) on confluence.

Following sections provides the differences between accessing the NetScaler LB and Big-IP via REST calls.

2.3.1. Access NetScaler LB via REST calls

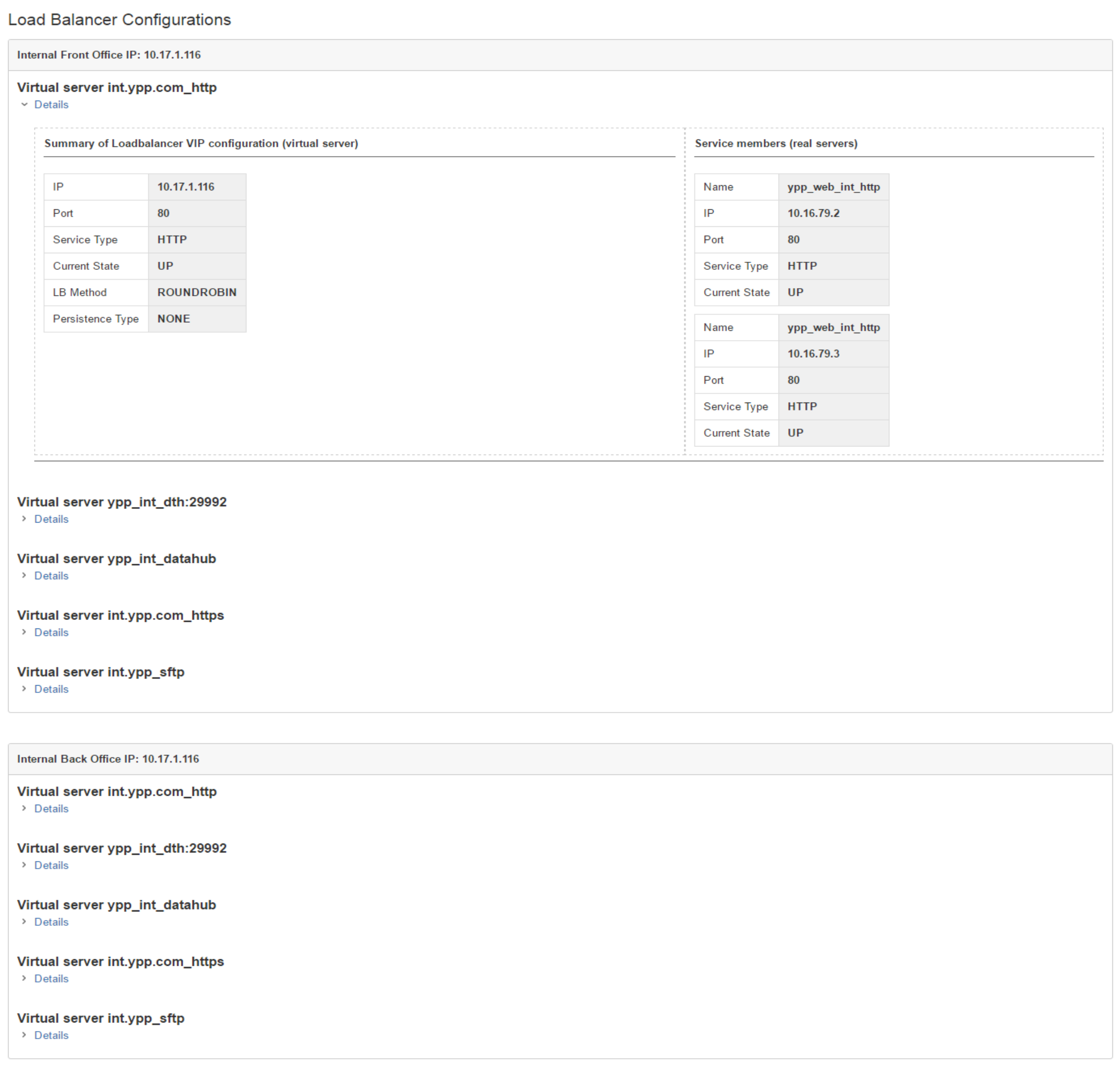
For accessing NetScaler LB, GET requests are always used and the access credential information is provided in the header of these GET requests. For accessing NetScaler LB, only access credentials were required, which means in order get data from NetScaler LB, stateless sessions were used that makes response time of getting the data from NetScaler LB smaller. NetScaler LB REST architecture designed by Citrix is capable of providing automatic functionality of filtering the data based on conditions, which is useful for filtering the configuration data based on front office IP or back office IP of customer environment [20].

2.3.2. Access Big-IP LB via REST calls

In order to access Big-IP LB, authentication token is required and for creating this authentication token POST request is used, where access credentials were provided in the headers of this POST request. Once the authentication token is created, this token will be valid for 1200 milliseconds and the same token is provided in the headers of GET request which are used to fetch the data from Big-IP LB. So, for every GET request there has to be a token created by POST request as the token is valid only for 1200 milliseconds. This means, in order to access the Big-IP LB sessions need to be maintained by creating this authentication token which makes response time of getting the data from Big-IP LB longer as compared to NetScaler LB. Big-IP LB REST architecture designed by F5 networks is not capable of providing automatic functionality of filtering the data based on conditions. Due to this, the logic of filtering the LB configuration data based on front office IP or back office IP of customer environment is implemented and this factor also makes longer response time of getting the data from Big-IP LB [21].

2.4. Results

In this chapter, the logic of REST API wrapper code of accessing the LBs via REST calls is explained in detail. After implementing this REST API wrapper code, it has been tested that the response time for getting all the required configuration data from both the load balancers is approximately 45 to 50 seconds which makes nightly cron jobs to publish the LB configuration data on confluence satisfactorily. Following is the sample LB configuration data of internal project environment generated from implemented REST API wrapper code:


Load Balancer Configurations

Internal Front Office IP: 10.17.1.116

Virtual server int.ypp.com_http
Details

Summary of Loadbalancer VIP configuration (virtual server)

IP	10.17.1.116
Port	80
Service Type	HTTP
Current State	UP
LB Method	ROUNDROBIN
Persistence Type	NONE

Service members (real servers)

Name	ypp_web_int_http
IP	10.16.79.2
Port	80
Service Type	HTTP
Current State	UP

Name	ypp_web_int_http
IP	10.16.79.3
Port	80
Service Type	HTTP
Current State	UP

Virtual server ypp_int_dth:29992
Details

Virtual server ypp_int_datahub
Details

Virtual server int.ypp.com_https
Details

Virtual server int.ypp_sftp
Details

Internal Back Office IP: 10.17.1.116

Virtual server int.ypp.com_http
Details

Virtual server ypp_int_dth:29992
Details

Virtual server ypp_int_datahub
Details

Virtual server int.ypp.com_https
Details

Virtual server int.ypp_sftp
Details

Figure 2.4-1 LB Configurations Data

In order to implement this REST API wrapper, cooperation of PQE team was required as their yTestRunner tool is used to publish the results on confluence, cooperation of sys admin team was required in order to understand the functionality of load balancers which is configured by them and to get the access credentials for load balancers. Finally, I as a part of PIE team implemented this REST API wrapper code in cooperation with PQE and sys admin teams. From this internship task, it can be seen that development and operations team are working hand in hand to continuously deliver the hybris e-commerce platform.

Chapter 3

Automation of Network Diagrams

3.1. Problem Statement

As mentioned in chapter 1, every customer project environment follows 3 tier architecture and every tier is a cluster of servers and for every customer environment there will be installation of hybris application. Since customer requires all the details of their project environments, previously dedicated resources in hCS team used to prepare the network diagram of all customer project environments by hand which is a time consuming, error prone manual work. It is a customer requirement to print the automated network diagram of all their environments as part of test scripts on confluence (wiki). PQE team had already automated this generation of network diagrams and publishing them on wiki using their yTestRunner tool. However, the generated diagrams were not using standard cisco icons [22] and in application tier of customer environment, there were cases where the customer environment required more servers which was difficult to fit in A4 sheet paper. These changes were requested by customer in the existing automated network diagrams implemented by PQE team and I, as a part of PIE team have implemented these changes in cooperation with PQE team.

3.2. Logic of Automated Network Diagrams

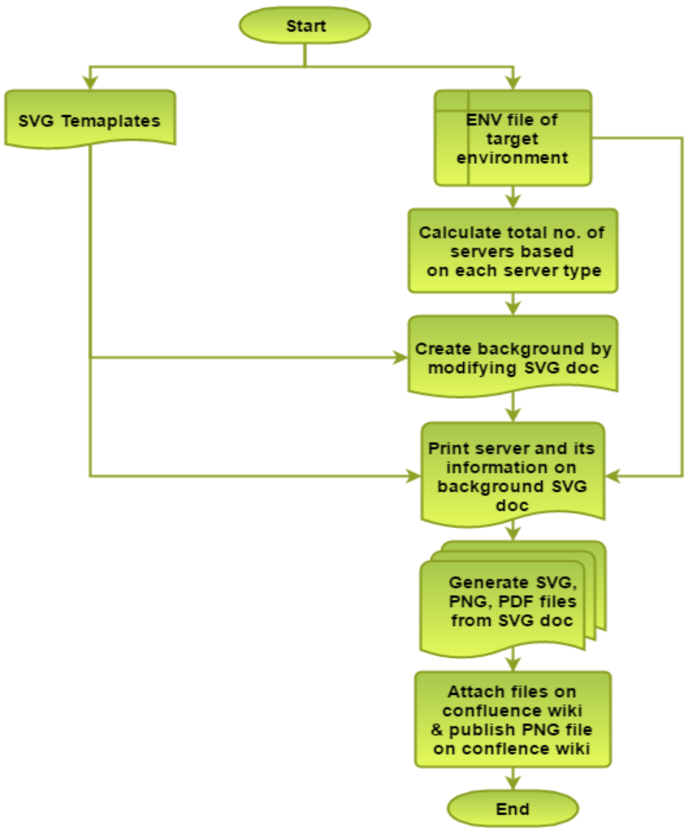

Start
SVG Temaplates
ENV file of target environment
Calculate total no. of servers based on each server type
Create background by modifying SVG doc
Print server and its information on background SVG doc
Generate SVG, PNG, PDF files from SVG doc
Attach files on confluence wiki & publish PNG file on conflence wiki
End

Figure 3.2-1 Automated Network Diagrams Flowchart

The above flowchart describes the basic steps performed to automate the network diagrams. For creating network diagrams, pre-designed svg templates were used and these svg templates are stored in the above enterprise stash repository URL. These svg templates are then modified based on the information of servers, domains, datacenters gathered from environment description file. Hence, as part of the internship task, following things in existing logic of automated network diagrams have been modified by me:

- SVG templates of server types as per standard ciso icons
- Implementing the logic of printing special server icons if there total number of servers of each server type (Admin, Web, Search, Datahub, etc.) are greater than 9 in single server type row based on the selected server row and server type.

For printing this network diagrams AWT (Abstract Window Toolkit) [23] and Apache Batik [24] libraries of java are used.

As mentioned earlier in chapter 2 that yTestRunner tool is maintained using enterprise stash repository system and git version control system. Thus, the implemented source code of generating automated network diagrams lies at following URL of enterprise stash repository:

https://enterprise-stash.hybris.com/projects/HCS-PDO-QE/repos/ytestrunner/browse/src/main/java/com/hybris/ms/qa/forms/svg

Following are the java classes used for generating the network diagrams and their usages.

Java Classes and their descriptions:

- **com.hybris.ms.qa.forms.svg.CreateNetworkDiagram.java**
  This class is implemented to generate the network diagram in svg format based on the information of servers, domains, datacenters gathered from environment description file and print this generated network diagram in .svg, .png and .pdf format.

- **com.hybris.ms.qa.definitions.Datacenters.java**
  This class is used to get datacenter type information from environment description file of target environment.

- **com.hybris.ms.qa.definitions.Servertypes.java**
  This class is used to define server type information of target environment.

- **com.hybris.ms.qa.testrunner.environment.Host.java**
  This class is used to get the server properties like name, RAM size, and server type from environment description file of target environment.

- **com.hybris.ms.qa.testrunner.environment.TargetEnvironment.java**
  This class is used to get the target environment ID of customer project.

- **com.hybris.ms.qa.testrunner.TestStep.java**
  This class is used to define abstract methods of defining test steps in test scripts and every class extending this class has to override these methods.

- **com.hybris.ms.qa.testrunner.drivers.documenter.ConfluenceDocumenter**
  This class is used to for confluence driver connection and publishing data on confluence document.

- **com.hybris.ms.qa.testrunner.environment.Hostname.java**
  This class is used to get domain name, IP address of host from environment description file of target environment.

- **com.hybris.ms.qa.testrunner.environment.TargetEnvironmentTools.java**
  This class is used to get the front office and back office IP of customer project based on the target environment ID of customer project.

3.3. Results

In this chapter, basic logic of automation of network diagrams is explained. After implementing the changes in the existing logic of automated network diagrams, enough testing on customer project environments is done and customer are satisfied with the generated network diagrams. Following is the sample of automated network diagram of internal project environment:

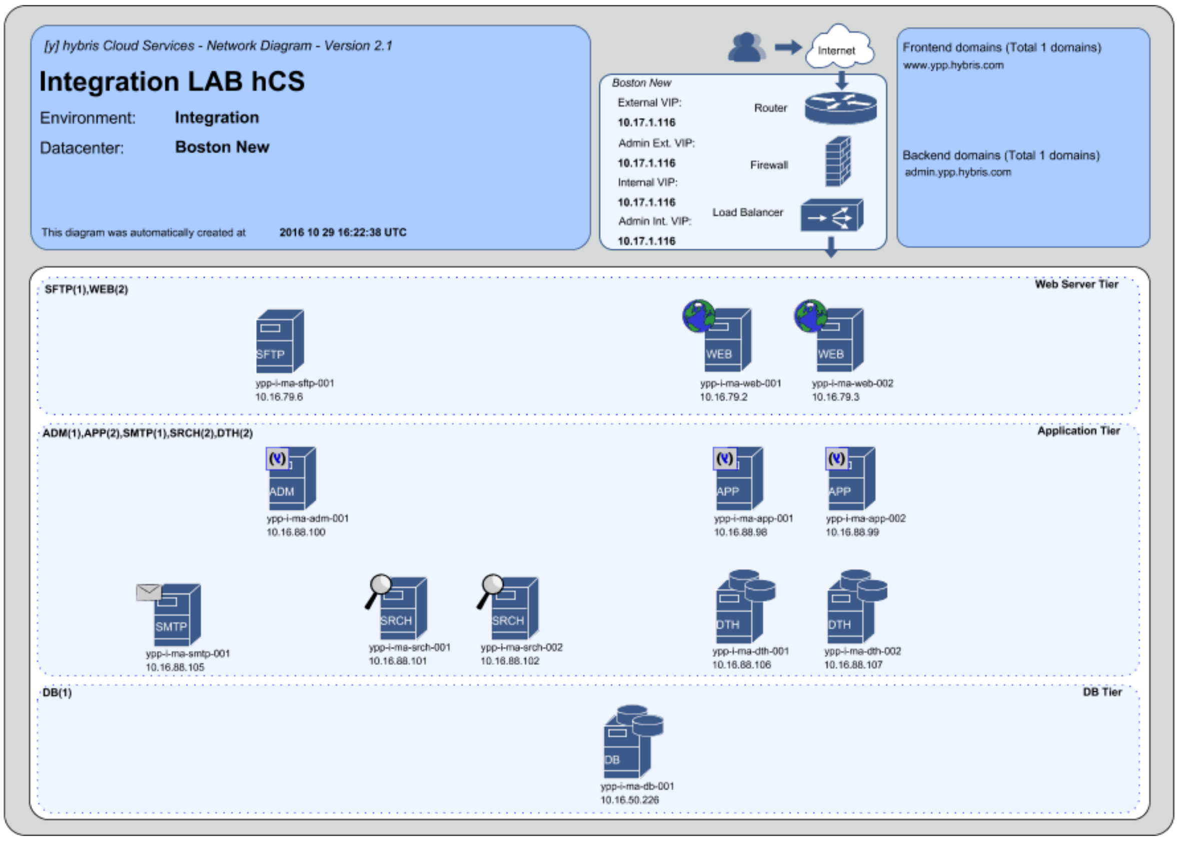


Figure 3.3-1 Automated Network Diagram

Chapter 4

Summary

Following are the valuable technical skills that are learnt during this internship period:

- Basic DevOps concepts.
- Basic Automation concepts.
- Java libraries for REST calls and SVG documents.
- Implementing test scripts using Java.
- Technical documentation system – confluence wiki.
- Enterprise stash repository system.
- Git version control system.
- Agile or scrum methodology.
- Service Now and Jira issue tracking systems

I would like to thank all respected **faculties of SRH Hochschule Heidelberg** for providing excellent knowledge and trainings in applied computer science master course which in turn helped me to get the Internship at SAP Hybris. I thank **Prof. Dr. Gerd Moeckel, Dean** of SRH Hochschule Heidelberg for his support and motivation for obtaining internship.

The overall experience at SAP Hybris during my internship was great in terms of learning new things, enhancing professional as well as technical skills. I would like to thank **Ionut Roibu (Platform Architecture and Automation – DevOps Manager)** for giving me opportunity to work as an intern at SAP Hybris, **Strahinja Stankovic (PIE Lead)** and **Andreas Zottmann (PQE Lead)** for providing immense knowledge and guidance, my friend and colleague **Pratiksha Jain (fellow working student)** for her support and all other SAP Hybris employees for making my work easier.

References

[1] "hybris: Omni-Channel Commerce Solution Overview | SAP Hybris," SAP Hybris, [Online]. Available: http://www.hybris.com/en/commerce.

[2] J. Wettinger, U. Breitenbücher and F. Leymann, "DevOpSlang – Bridging the Gap between Development and Operations," in *Service-Oriented and Cloud Computing*, University of Stuttgart, Germany, Springer Berlin Heidelberg, 2014, pp. pp 108-122.

[3] Amel Badaoui, "Tech Area Core - Rolling Update Trail," 11 08 2016. [Online]. Available: https://wiki.hybris.com/display/~amel.badaoui@hybris.com/Tech+Area+Core+-+Rolling+Update+Trail?preview=%2F311521226%2F311521208%2Fhybris+cake+-+platform.png.

[4] Katarzyna Kepka, "Introduction to Customers and their System Landscape," 20 05 2016. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/Introduction+to+customers+and+their+system+landscape.

[5] Kamil Czelen, "PIPS configuration standards - Environments," 17 03 2015. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/PIPS+configuration+standards+-+Environments.

[6] Dan Chiasson, „PIE training plan - [y] hCS Platform DevOps - hybris Wiki," 29 09 2015. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/PIE+training+plan?preview=%2F287504736%2F287504851%2FPIE+Team+hybris+Training.pptx.

[7] Maria Krings, "hCS Hosting Models," 24 01 2014. [Online]. Available: https://wiki.hybris.com/display/MSS/hCS+Hosting+Models.

[8] Amel Badaoui, "hCS default Infrastructure Setup," 08 03 2016. [Online]. Available: https://wiki.hybris.com/display/prodsup/hCS+default+Infrastructure+Setup?preview=%2F299643832%2F299647264%2FSupport+-+Data+Center+basics.pptx.

[9] Dennis Benzinger, "hybris System Administrators Training Material," 10 09 2012. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/hybris+System+Administrators+Training+Material.

[10] Dan Chiasson, "Platform Implementation Engineering," 18 12 2013. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/Platform+Implementation+Engineering.

[11] Maciej Tosza, "Platform Service Engineering," 03 09 2013. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/Platform+Service+Engineering.

[12] Andreas Zottmann, "Platform Quality Engineering," 08 05 2013. [Online]. Available: https://wiki.hybris.com/display/MSPIPS/Platform+Quality+Engineering.

[13] Dennis Benzinger, "Platform Monitoring Engineering (PME)," 21 08 2015. [Online]. Available: https://wiki.hybris.com/pages/viewpage.action?pageId=284733407.

[14] "Load Balancing - Citrix," Citrix, [Online]. Available: https://www.citrix.com/glossary/load-balancing.html. [Accessed 09 2016].

[15] "Load Balancer," f5, [Online]. Available: https://f5.com/glossary/load-balancer. [Accessed 09 2016].

[16] "NITRO API," Citrix, [Online]. Available: http://docs.citrix.com/en-us/netscaler/11/nitro-api.html. [Accessed 09 2016].

[17] "iControl REST Home - DevCentral Wiki," f5, [Online]. Available: https://devcentral.f5.com/wiki/iControlREST.HomePage.ashx. [Accessed 09 2016].

[18] "Apache Commons," [Online]. Available: https://commons.apache.org/. [Accessed 08 2016].

[19] Strahinja Stankovic, "LB API Access," [Online]. Available: https://wiki.hybris.com/display/MSPIPS/LB+API+Access. [Accessed 08 2016].

[20] Strahinja Stankovic, "NetScaler LB REST API," [Online]. Available: https://wiki.hybris.com/display/MSPIPS/NetScaler+LB+REST+API. [Accessed 09 2016].

[21] Alpana Chaphalkar, "F5 LB REST API," [Online]. Available: https://wiki.hybris.com/display/MSPIPS/F5+LB+REST+API. [Accessed 09 2016].

[22] "Network Topology Icons - Doing Business With Cisco - Cisco," [Online]. Available: http://www.cisco.com/c/en/us/about/brand-center/network-topology-icons.html. [Accessed 08 2016].

[23] „java.awt Java Platform SE 7,“ Oracle, [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html. [Zugriff am 08 2016].

[24] „Using Batik,“ apache.org, [Online]. Available: http://xmlgraphics.apache.org/batik/using/. [Zugriff am 08 2016].