Morpheus Instance Plugins for Monitoring XaaS

Tanya Dinesh

Student

Department of Information Science and Engineering

RV College of Engineering

Abstract: *Anything as a service or XaaS in short is a general term which encompasses all the services related to cloud computing or remote access. While setting up cloud infrastructures, that hosts theses services there are a lot of parameters that need maintained and configured, these parameters can be automated with the help of Ansible, an automation platform. But displaying these configured resources to the user is equally as important. This work focuses on creating XaaS plugins for the Morpheus cloud hybridization platform to monitor the current configuration of the deployed resources. The plugins fetches live data from the CMDB asynchronously using API calls through a wrapper API. The developed project is successful in meeting the requirements set out.*

Keywords: XaaS, CMDB, Wrapper API, Morpheus Plugins

1. INTRODUCTION

Anything as a service describes a general category of services related to cloud computing and remote access. It includes a huge number of technologies, and products which is made accessible to the user via the internet. Basically, any kind of information technology function can be adapted to be served via the internet as a service. This kind of service is usually more flexible as it set up based on a flexible subscription model rather than as a one-time upfront purchase. In this work, Ansible and i-doit CMDB is used for provisioning and configuring the resources. Configuration management database or CMDB for short is a database that stores arrays of information about information and relationships [1]. It provides a common storage place for all information about information technology or IT assets and other configuration items for authorized people to access.

Morpheus is a hybrid cloud automation platform, designed from the ground up to unify management of multi-cloud and hybrid IT while empowering DevOps teams with self-service provisioning of bare metal, VM, and container-based application services. Morpheus plugins are compiled jar files which are loaded onto the website to create custom interfaces for the user which can show and perform various actions. API wrappers simplify the process of interacting with APIs [2]. An API wrapper provides a way to access an API through a particular programming language or interface, which can help streamline the process of making API calls.

The main motivation behind the project and cloud computing is to enable businesses to get access to data centers and manage tasks from a remote location. Cloud computing works on the pay-as-you-go pricing model, which helps businesses lower their operating cost and run infrastructure more efficiently. Creating plugins will stream line the process of using or configuring the cloud infrastructure and make it more convenient to use infrastructure.

There are two main objectives with this project, one is to create instance plugins for the Morpheus dashboard which displays the current status and configuration of the XaaS services provisioned. And the second objective is to create a wrapper API for CMDB to add a level of abstraction and support further enhancements.

2. LITERATURE SURVEY

Groovy is chosen for the backend for the plugins and the test conducted on Groovy service shows that the overhead of Groovy service is acceptable for a flexible service

framework [3]. It improves the service performance for mass data processing by reducing the data transmission on network. The Groovy service is useful and necessary for the improving web service for users on all levels. Morpheus is the hybrid cloud management system which acts as the backbone behind the project. The Anisible playbooks and the CMDB is all linked to Morpheus. Morpheus uses a configured instance of a workload, annotated with the configuration information, and automatically configures new instances of the workload [4]. One of the leading platforms for deploying and deploying XaaS services.

XaaS or anything as a service falls under the cloud services domain, there are many benefits to using cloud services rather than the traditional services [5]. Provisioning resources dynamically using heuristics is used to intelligently allocate resources to keep the cost of the whole infrastructure down [6]. Morpheus is a hybrid cloud platform that can allocate resources from various sources and manage them on the fly with the help of deep integration with Ansible playbooks. There are a lot of benefits of Continuous Integration and Continuous Deployment Pipeline automation and the use of Ansible, it helps save on a lot of resources [7]. UI design of the plugin is very reminiscent of the design principles used in building a dashboard. M. Holjevac [8] discusses and compares various UI templates for different applications. In the project, custom templates is used to create user interfaces according to the specifications set by the client.
The project follows agile methodology for its development, this is done for a myriad or reasons stated below. Five of these SDLC or system development life cycle models, including the waterfall model, spiral model, V-shaped model, iterative model and prototype model are included in for comparison with other models [9]. A study determined that agile ceremony based approach resulted in a significant increase in sprint velocity by over 13% [10].

3. ARCHITECTURE

The plugins are designed to be integrated with an existing cloud infrastructure. Hence, the data to be displayed is to be fetched from the CMDB. Fig 1 depicts the High-Level design of the plugins for XaaS. It comprises of three main blocks, namely Ansible, Python API wrapper and i-Doit CMDB. Ansible [11] with its Playbooks and Roles is used to configure the XaaS services, Python CMDB acts as an intermediary between the CMDB and Ansible playbooks. I-Doit CMDB is the database where the configuration of all the deployed services is stored. I-doit CMDB has its own API interface but this is not used by the plugins, instead a wrapper API created with Python and the FastAPI library is used.

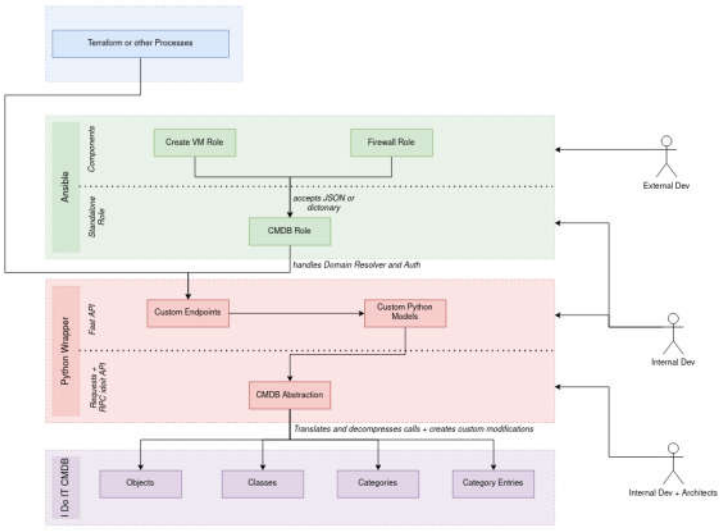

Fig 1: High Level Design of the XaaS Plugins

In Fig. 2 the sequence of operation for fetching the data from CMDB is depicted. The logged in user selects the instance that they want to access, this request is handled by Morpheus and it triggers the plugin related to that particular instance.

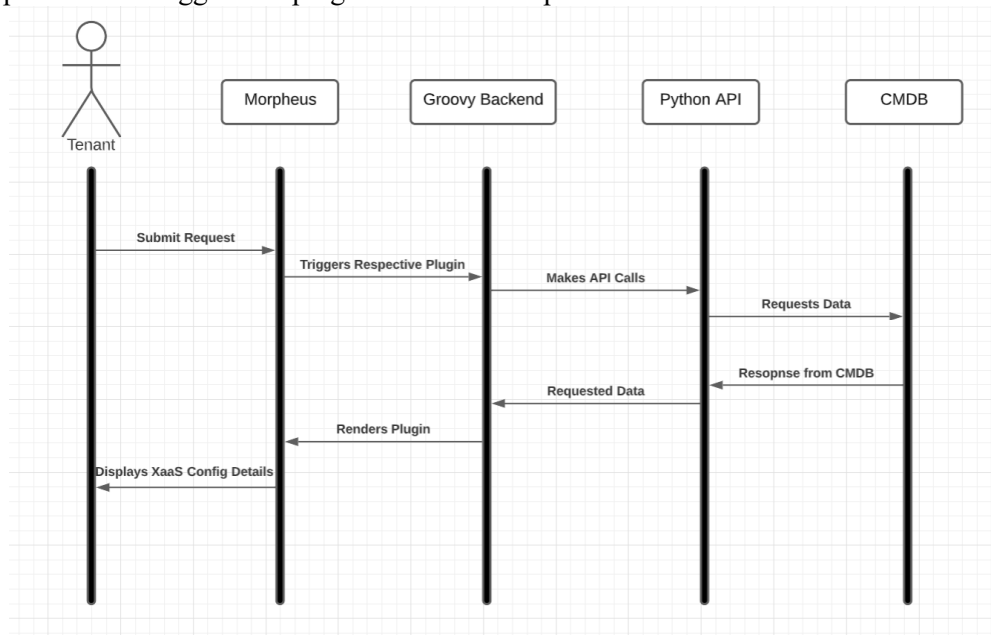

Fig: 2 Sequence Diagram of a Plugin Fetching Data from CMDB

4. METHODOLOGY

4.1. Templates

The visible part of the plugins is rendered by templates. These consists of HTML CSS and JavaScript files. Development of these files were done using static JSON files that were acquired from the responses of several API calls to the CMDB. JQuery data tables are used for displaying data that is in tabular form [12]. Datatables support fetching data asynchronously using AJAX. EvoTables the open source calendar library for JavaScript is used in the backup XaaS plugin. Figure 3 depicts the UI of the Firewall template.

No front-end frameworks such as React or Angular is used as the plugins are not capable of running another framework on top of the framework used in Morpheus website. Hence, native JavaScript is used for the templates of all these plugins.

4.2 Wrapper API

i-Doit CMDB has its own API interface for accessing the interface of various XaaS services. However this native API is not used by the plugins. This is mainly due to 2 reasons, one, there is no pagination support in the native CMDB API. In production each XaaS services might have thousands if not tens of thousands of records. Processing and rendering all these records in the front end is extremely resource intensive. Hence, server side processing is used with pagination support from the API to enable asynchronously pulling data from CMDB in smaller chunks. Second, in the future when the i-doit CMDB is updated and the interface changes the only thing that needs to be changed is the wrapper API and not wherever the API is called in the plugins [13]. Fig 4 showcases the pagination endpoint in the API for DNS records in Swagger UI.

4.3 Groovy Backend

Groovy along with Gradle [14], a build automaton tool is used to create the backend of these plugins. It served as a container to the templates the render the UI for different plugins. The logic for user authentication, instance selection and rendering the templates is handled by Groovy. Gradle is used to automate the build process of the plugins and to ensure that the plugins are up to specifications.

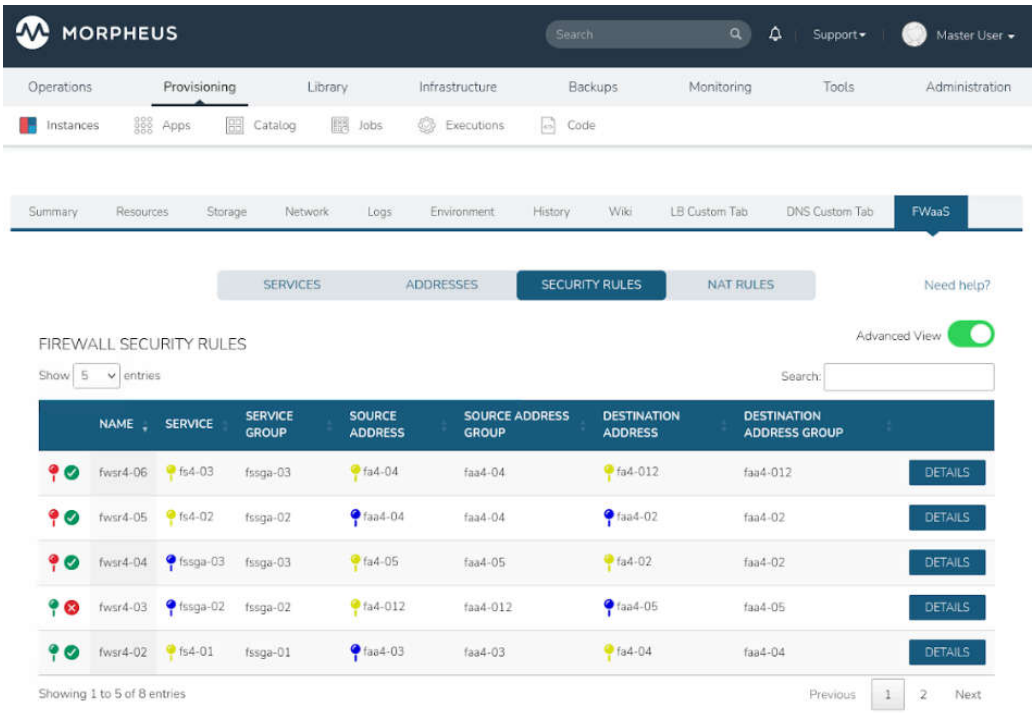

Fig: 3 Security Rules Tab of the FW Plugin

5. TESTING

Software testing for the project is done in every stage of development to ensure that all the components are up to standard and are functioning as expected. In case of the plugins, the templates is tested locally using static data. The modifications to the wrapper API is done and tested with Postman [15], to ensure all the API endpoints are accessible and are showing the correct responses. This is then followed with integrating the template into a plugin with groovy backend. The 5 plugins are tested individually with their respective templates and API calls. The plugins are combined into one plugin which serves and renders required template. Error logs in Morpheus is used to debug and sort out any of the errors and discrepancies encountered during testing. Unit testing in the project involves testing the various plugin templates with static data. This static data is fetched directly from CMDB in the form of JSON arrays, and hence will be representative of the type of data which it will receive in production. In integration testing the UI templates are linked to the Groovy backend using a generic test plugin. The same test plugin is used to test all the 5 plugin templates. All the 5 XaaS plugins are combined to create one plugin and one jar file. The plugins are rendered based on the type of instance selected by the user. All the tests were successful and the plugin performs as expected in all scenarios.

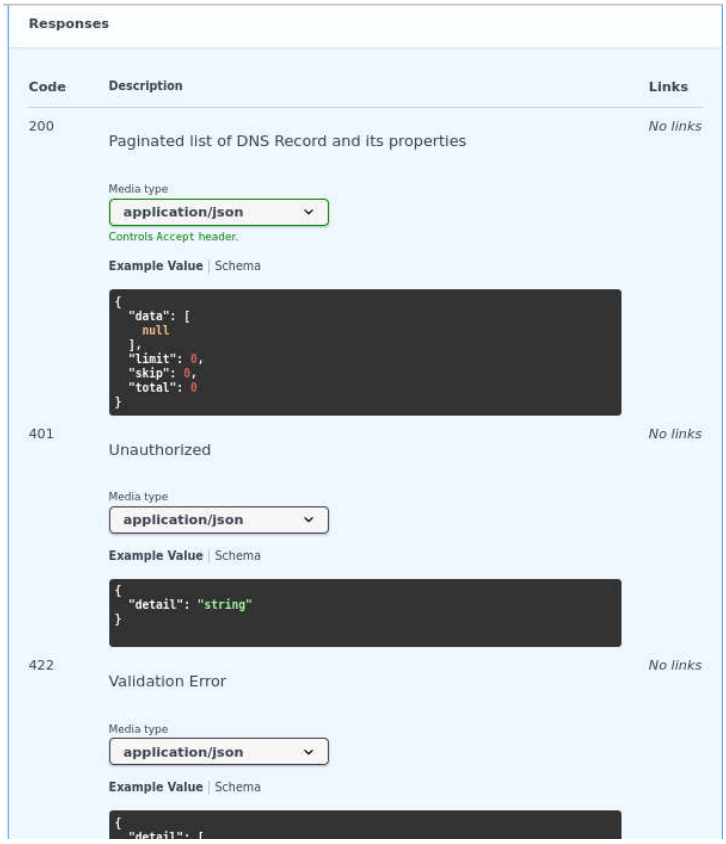

Fig: 4 API Endpoint for Paginated DNS Records

6. CONCLUSION AND FUTURE SCOPE

The built plugins meet all the functional and non-functional requirements set at the beginning of the project. The wrapper API for the CMDB is enhanced to support pagination to support server side processing with jQuery datatables. These plugins act a front end to a large cloud infrastructure built with servers and networking infrastructure. The system is successful in displaying the current configuration of the various cloud services deployed i.e. DNS, Firewall, Load Balancer, S3 and Backup XaaS. These services are deployed and configured by Ansible playbooks and roles, i-doit CMDB and Morpheus cloud management and automation system. The plugins display the instances of services only linked to the logged in tenant. The plugins asynchronously refreshed its data every 30 seconds so as to provide the most up to date data from the CMDB.

There are a few limitations in the current version of the plugins that can be improved in the future versions. These include adding mobile browser support, adding support to edit catalog items directly from the plugin, instead of being redirected and using an alternative solution for the CMDB database for better performance.

REFERENCES

[1] *M. Brenner and M. Gillmeister, "Designing CMDB data models with good utility and limited complexity," 2014 IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1-15, doi: 10.1109/NOMS.2014.6838375.*

[2] *Y. Park and D. Min, "Development of HLA-DDS wrapper API for network-controllable distributed simulation," 2013 7th International Conference on Application of Information and Communication Technologies, 2013, pp. 1-5, doi: 10.1109/ICAICT.2013.6722799.*

[3] *ZhenChun Huang and Chuan He, "Groovy service: on-demand Web service by script language," IEEE International Workshop on Service-Oriented System Engineering (SOSE'05), 2005, pp. 105-110, doi: 10.1109/SOSE.2005.13.*

[4] *D. Jeswani, R. Balani, A. Verma and K. Bhattacharya, "Morpheus: Learning configurations by example," 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 76-84.*

[5] *Y. Duan, Y. Cao and X. Sun, "Various "aaS" of everything as a service," 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015, pp. 1-6, doi: 10.1109/SNPD.2015.7176215.*

[6] *Z. Cai, X. Li and J. N. D. Gupta, "Heuristics for Provisioning Services to Workflows in XaaS Clouds," in IEEE Transactions on Services Computing, vol. 9, no. 2, pp. 250-263, 1 March-April 2016, doi: 10.1109/TSC.2014.2361320.*

[7] *S. Mysari and V. Bejgam, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-4, doi: 10.1109/ic-ETITE47903.2020.239.*

[8] *M. Holjevac and T. Jakopec, "Web application dashboards as a tool for data visualization and enrichment," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), 2020, pp. 1740-1745, doi: 10.23919/MIPRO48935.2020.9245289.*

[9] *A. Sinha and P. Das, "Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry," 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), 2021, pp. 1-4, doi: 10.1109/IEMENTech53263.2021.9614779.*

[10] *S. Sharma, D. Kumar and M. E. Fayad, "An Impact Assessment of Agile Ceremonies on Sprint Velocity Under Agile Software Development," 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2021, pp. 1-5, doi: 10.1109/ICRITO51393.2021.9596508.*

[11] *N. Jain, H. Shrivastava and A. A. Moghe, "Production-ready environment for HLS Player using FFmpeg with automation on S3 Bucket using Ansible," 2nd International Conference on Data, Engineering and Applications (IDEA), 2020, pp. 1-4, doi: 10.1109/IDEA49133.2020.9170694.*

[12] *A. Hume, N. Ferreira and L. Cernuzzi, "The design of a privacy dashboard for an academic environment based on participatory design," 2021 XLVII Latin American Computing Conference (CLEI), 2021, pp. 1-10, doi: 10.1109/CLEI53233.2021.9640155.*

[13] *L. Li, W. Chou, W. Zhou and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," in IEEE Transactions on Network and Service Management, vol. 13, no. 1, pp. 154-167, March 2016, doi: 10.1109/TNSM.2016.2516946.*

[14] *C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, "DevOps," in IEEE Software, vol. 33, no. 3, pp. 94-100, May-June 2016, doi: 10.1109/MS.2016.68.*

[15] *R. M. Poston and M. P. Sexton, "Evaluating and selecting testing tools," [1992] Proceedings of the Second Symposium on Assessment of Quality Software Development Tools, 1992, pp. 55-64, doi: 10.1109/AQSDT.1992.205836.*