PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené ako otvorená cloudová služba s prvkami strojového učenia

# Previazané s balíkom KPB6 – Publikačné výstupy

Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD VLÁDY
SLOVENSKEJ REPUBLIKY

# VoIP as a Service: Asterisk in OpenStack platform

Martin Kontsek*, Pavel Segec, Marek Moravcik, Ivana Bridova
Faculty of Management Science and Informatics, University of Zilina, Univerzitna 8215/1, 010 26 Zilina
*e-mail: martin.kontsek@fri.uniza.sk

***Abstract*—This article explores the integration of Asterisk as a VoIP service within the OpenStack cloud platform. It highlights the advantages of cloud services, particularly for end users and IT departments. The implementation process of Asterisk, including installation, configuration, and the development of automation scripts, is detailed. The significance of Network Address Translation (NAT) functionality is emphasized, along with the development of user-friendly configuration scripts. Lastly, the article discusses challenges encountered with the ManageIQ integration, suggesting potential improvements for future deployments.**

***Index Terms*—Cloud Computing, OpenStack, Asterisk, VoIP, ManageIQ**

## I. Introduction

The trend toward the utilization of cloud services is currently gaining momentum. This phenomenon is driven by various factors, including technological advancements as well as economic and organizational aspects. For end users, leveraging cloud services presents a straightforward and efficient means of accessing diverse applications and data storage without the need to invest in personal infrastructure. Financially, it often proves to be more advantageous, as companies can avoid the high costs associated with purchasing expensive servers and hardware, opting instead to use available cloud platforms for a monthly fee.

For administrators and IT departments, utilizing cloud services is equally beneficial. Managing the devices on which these services operate becomes simpler, as physical maintenance and server updates are no longer a concern. Cloud platforms often provide tools for monitoring and managing services, enhancing efficiency and offering better insights into who utilizes the service, when, and to what extent.

In the context of our specific task—implementing Asterisk as a cloud service for the staff of the Department of Information Networks—it is essential to consider multiple aspects. These include security, scalability, availability, and performance. The implementation of Asterisk in the cloud should be well thought out, taking into account user needs as well as the technical requirements of the system. It is crucial to ensure that communication through Asterisk is reliable and secure while also being sufficiently flexible to adapt to the changing needs of users. The selection of an appropriate cloud platform and the configuration of Asterisk are key steps in this implementation.

## II. Asterisk

Asterisk is an open-source software framework designed for building multi-protocol communication applications in a real time, primarily focused on voice over IP (VoIP) services. Initially developed by Mark Spencer in 1999 and now maintained by Sangoma Technologies, Asterisk has evolved into one of the most versatile and widely adopted platforms for creating telephony applications. It is used globally to support a range of services, including private branch exchange (PBX) systems, VoIP gateways, and conference servers. Asterisk's open-source nature provides users with the flexibility to develop tailored communication solutions, whether for small business environments or large-scale telephony infrastructures. As a robust and cost-effective alternative to proprietary telecommunication systems, Asterisk has gained widespread use in both enterprise and service provider markets. Asterisk is used by various states all around the world.

One of the primary strengths of Asterisk lies in its adaptability, as it supports a wide array of communication protocols, including the Session Initiation Protocol (SIP), H.323, and other legacy telephony systems. This flexibility allows seamless integration with both traditional and IP-based telephony networks, providing businesses with the opportunity to modernize their communication infrastructure without fully abandoning older technologies.

### A. Key features

Asterisk provides a comprehensive set of features that to meet both basic and advanced communication needs. At its core, Asterisk functions as a fully-featured PBX (Private Branch Exchange) system, offering essential call management features such as call routing, voicemail, call transfer, and call waiting. These capabilities allow businesses to effectively manage internal and external communication, optimizing their workflow through automation and streamlined call handling processes. Additionally, Asterisk's PBX functionality can be enhanced by integrating it with Interactive Voice Response (IVR) systems, which enable the automation of incoming call handling through voice menus.

One of Asterisk's defining features is its role as a VoIP gateway, bridging traditional telephony systems with modern VoIP networks. By supporting multiple communication protocols, Asterisk allows organizations to transition from analog and digital telephony systems to VoIP, enabling them to leverage the cost efficiency and scalability of internet-based communications. The inclusion of industry-standard protocols

such as SIP and H.323 ensures compatibility with a wide range of VoIP services and devices, while also allowing the integration of legacy systems via MGCP and SCCP.

Asterisk's dialplan configuration system further enhances its flexibility by allowing users to define custom call handling logic. The dialplan is highly configurable and can be scripted to manage call routing based on user-defined rules, which can be as simple or complex as required [1, 2].

### B. Architecture

Asterisk's architecture is composed of a core system that manages key functionalities, such as reading configuration files and building the dialplan, which controls how calls are handled. The core interacts with modules, which add various capabilities, including channel drivers (e.g., SIP), voicemail, and conferencing features. These modules are dynamically loaded, allowing for customization and scalability. Channels facilitate communication between devices, and calls are managed by connecting these channels. Asterisk's architecture is modular, enabling integration with other telephony systems and extensive customization [3].

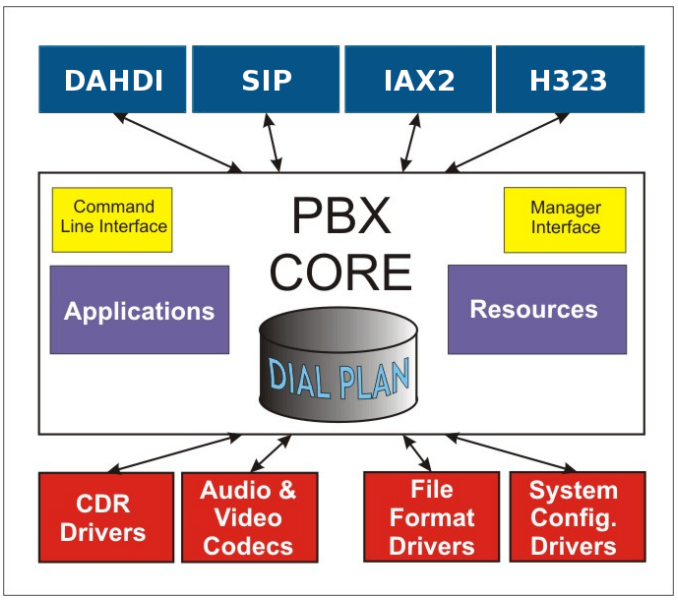


Fig. 1. Asterisk Architecture [4]

### C. Difference between chan_sip and pjsip

In Asterisk, the chan_sip and PJSIP modules serve as SIP channel drivers, but they differ significantly in terms of architecture, performance, and features. chan_sip, the older and now deprecated module, was widely used for basic SIP functionality, but it faced limitations in terms of scalability, security, and complex configuration management. PJSIP, by contrast, is a modern, highly scalable SIP stack that offers enhanced performance, modularity, and better support for advanced features like NAT traversal, TLS, and SRTP. Unlike chan_sip, PJSIP also supports asynchronous operations, which improve system efficiency and responsiveness under high call volumes. Additionally, PJSIP's configuration is more flexible and allows for better integration with external systems, making it the preferred choice for contemporary Asterisk deployments. Due to its ongoing support and richer feature set, PJSIP is generally recommended for new Asterisk installations over chan_sip, which has been deprecated [5].

## III. OpenStack

OpenStack is an open-source cloud computing platform designed to enable the development and management of public and private clouds. Initially developed by NASA and Rackspace, OpenStack is now maintained by the OpenStack Foundation. It provides a set of interrelated services that manage computing, storage, and networking resources through a web-based dashboard, command-line tools, and RESTful APIs. OpenStack is designed to be highly scalable, allowing businesses to build large, multi-tenant cloud infrastructures.

### A. Key features

One of OpenStack's primary features is its modular architecture, which is composed of various services that can be integrated based on an organization's needs. These services include Nova, which provides compute resources by managing virtual machines (VMs), and Neutron, responsible for managing networking, enabling complex network configurations such as software-defined networking (SDN). Another core service is Cinder, which handles block storage, enabling the creation and management of persistent storage volumes for instances. For object storage, OpenStack offers Swift, which provides scalable storage ideal for archiving and large-scale data storage.

OpenStack also emphasizes multi-tenancy, allowing different users or departments within an organization to share a cloud environment while maintaining isolated resources. This capability makes it ideal for both private and public clouds, where resource allocation and security are paramount. Furthermore, OpenStack is highly scalable and can be deployed across hundreds or even thousands of servers, making it suitable for large-scale enterprise environments. Its open-source nature and large community of contributors ensure that it stays at the forefront of cloud innovations while offering a cost-effective solution compared to proprietary cloud services. OpenStack's flexibility, wide-ranging support for virtualization technologies, and active community contribute to its growing adoption in both enterprise and service provider environments [6].

## IV. Integration of Asterisk to OpenStack

### A. Installation and configuration of Asterisk

The initial phase of our project involved the manual installation of Asterisk on a Debian instance within the OpenStack environment. We began by deploying a new instance featuring the latest version of Debian. Our intention was to install Asterisk directly from the default package repositories. However, we encountered an obstacle, as the version of Asterisk available was outdated and not aligned with our requirements.

To address this issue, we needed to configure the system to include access to older repositories by adding the necessary entries to the sources list.


Fig. 2. Debian sources list

Upon successfully installing Asterisk, we developed a basic configuration that enabled calls to a specific number, accompanied by an audio playback feature. This initial setup facilitated our understanding of Asterisk's capabilities. We then enhanced the configuration to allow for calling between various users registered within the Asterisk exchange. These configurations served as valuable exercises for reinforcing our knowledge of Asterisk's functionalities. However, recognizing the importance of utilizing updated technology, we decided to upgrade Asterisk to a newer version to incorporate the res_pjsip module, which is a modern alternative to the deprecated chan_sip module.

To facilitate this transition, we terminated the existing Debian instance and launched a new one, following the installation guidelines specified in the Asterisk documentation [7]. Once the installation was complete, we created a configuration nearly identical to our previous setup, ensuring it was compatible with the res_pjsip module. Given the lengthy nature of the testing phase, we opted to develop a script that could be embedded in the instance's metadata during its creation. This script would automate the installation of all prerequisites for Asterisk, streamline the installation process itself, and establish a fundamental test configuration.

### *B. NAT Functionality*

The subsequent step in our project involved the integration and testing of Network Address Translation (NAT) functionality, which is critical for VoIP applications operating within private networks. To implement this, we utilized the Floating IP feature provided by OpenStack. We initiated the creation of a new private network, which enabled the addition of new instances while ensuring connectivity to the public internet via a router. By assigning the Floating IP to the external interface of the router, we ensured that our VoIP exchange became accessible from the public network, thus facilitating communication with external devices beyond the confines of the private network.

### *C. Script for creating configuration*

In our efforts to streamline the configuration process, we aimed to provide users with an accessible method for configuring the exchange without requiring in-depth knowledge of Asterisk module configuration files. To this end, we developed a user-friendly script designed to generate a basic configuration and assist users in creating account configurations for the exchange. The script requires minimal user input, specifically the names and passwords for the accounts to be created. Users can specify one or multiple accounts during the initial setup, and the script also allows for the generation of new configurations or the addition of additional accounts as needed.

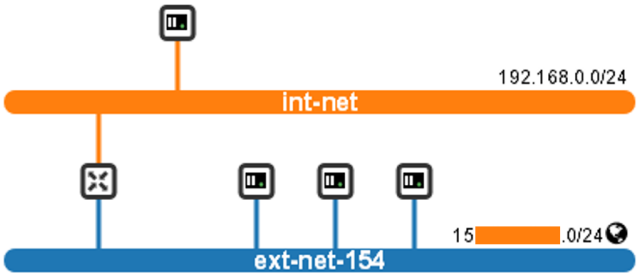


Fig. 3. OpenStack network with FloatingIP router

Finally, we sought to further automate the entire process through ManageIQ, enabling users to deploy new instances with a single click. However, this endeavor proved impractical, as the implementation of ManageIQ in our OpenStack environment did not incorporate all the desired features, limiting the automation capabilities we had envisioned.

## V. Conclusion

In this article, we presented an overview of Asterisk and OpenStack, highlighting their key features and architectures. We discussed our efforts to implement Asterisk as a VoIP solution offered as a service. Initially, we attempted to create a modified Debian image with a preinstalled version of Asterisk but encountered issues, including problems with unauthorized packages and installation errors related to library dependencies. Consequently, we opted for a simpler approach, utilizing a clean Debian image and automation scripts for installation and configuration.

The script we developed checks prerequisites and generates a basic configuration for Asterisk. Users can easily add one or more accounts by inputting the relevant names and passwords, and this script can be executed at any time.

Finally, we aimed to enhance the process further by integrating ManageIQ for automated instance deployment. Unfortunately, due to improper installation within our OpenStack environment, we could not deploy or test ManageIQ. However, we believe that, once correctly installed, it should function as intended.

## References

[1] *Getting Started with Asterisk*. URL: https://www.asterisk.org/get-started/.

[2] *Home - This is the home of the official documentation for The Asterisk Project*. URL: https://docs.asterisk.org/.
[3] *Asterisk Architecture*. URL: https://docs.asterisk.org/Fundamentals/Asterisk-Architecture/Asterisk-Architecture-The-Big-Picture/.
[4] *Asterisk docs - a big picture of Asterisk Architecture*. URL: https://docs.asterisk.org/Fundamentals/Asterisk-Architecture/bigpicture.png.
[5] *Asterisk docs - Migrating from chan$_s iptores_p jsip$*. URL: https://docs.asterisk.org/Configuration/Channel-Drivers/SIP/Configuring-res_pjsip/Migrating-from-chan_sip-to-res_pjsip/.
[6] *What is Openstack*. URL: https://www.openstack.org/software/.
[7] *Installing Asterisk From Source*. URL: https://docs.asterisk.org/Getting-Started/Installing-Asterisk/Installing-Asterisk-From-Source/What-to-Download/.