International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Implementation of FaaS in OpenStack Cloud

Rajeshwar Mukund and Prof. Rajesh Bharati

Department Of Computer Engineering Dr. D. Y. Patil Institute of Technology, Pimpri Pune, India

Received 10 Nov 2020, Accepted 10 Dec 2020, Available online 01 Feb 2021, **Special Issue-8 (Feb 2021)**

Abstract

With the evolution of Cloud Computing and it's use as a cloud user, Could Computing can be consider as various services in different aspects such as IaaS - Infrastructure as a Service, PaaS - Platform as a Service, SaaS - Software as a Service, Faas - Function as a Service As a cloud, FaaS user just need to write processing logic for function in available coding language supported by FaaS. Deployed function with processing logic in FaaS, it is ready to serve the purpose. Lifecycle for the function is until the execution of the FaaS. Function execution results does not stored on its own. Unlike traditional instance in cloud, FaaS is stateless. In order to maintain state FaaS can use supporting cloud services like object storage, no-sequel DB, DBs, parameter store etc. FaaS is well suited for computing context of lightweight computing, stateless / light state computing is required. This goes hand in hand with Micro service architecture and IOT platform. FaaS – Function as a Service is available in Public Hosted Cloud such as AWS, Microsoft Azure, Google Cloud, IBM Cloud etc. and Private Hosted Cloud Implementation such as OpenStack Cloud Implementation.

Keywords: *Cloud Computing; FaaS; Serverless; Apache Whisk; Fission; Iron Functions; Fn Project; Open Lambda; Kuberless; Open FaaS*

Introduction

In terms of Cloud Services layered architectural perspective, Cloud Computing Services classified based on who will manage the various layers of these services, as IaaS, PaaS, SaaS, & FaaS. In context of IaaS - Infrastructure as a Service in Cloud Computing, cloud provider manages underneath hardware & virtualization layer, which includes hardware servers, storage & networking. End user manages virtual instance, OS, Application, Availability, Scalability of Applications built on top of Infrastructure built on top of IaaS. In context of PaaS - Platform as a Service in Cloud Computing, cloud provider provides platform, which manages the virtual instances, OS, Availability & Scalability of instances build on top of IaaS. End user manages the data and applications built on top of SaaS. E.g. of SaaS is Google App Engine, which allows applications to be run on Google's infrastructure, and Salesforce's Force.com platform. Sahara service in Open Stack cloud which creates and manages underneath Hadoop Cluster. In context of SaaS - Software as a Service in Cloud Computing, cloud provides manages all aspects of applications from Infrastructure, OS, Availability, Scalability, Platform, Applications & Data as well. IaaS as well as FaaS used as a base for building SaaS, examples of SaaS are Microsoft Office 365, Oracle Cloud database offering.

In the context of FaaS - Function as a Service in Cloud Computing, cloud consumer neither care about the underneath Infrastructure on which the code is running nor the availability of underneath components. In traditional approach of Cloud Computing, an instance to be spawn on the VPC with certain amount of RAM, Computing cores & Storage attached to instance. Post instance availability and configuration, instance is available to serve the intended purpose for the instance.

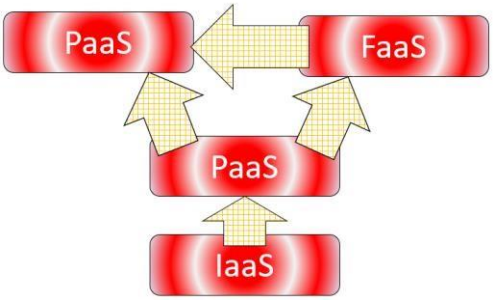

Fig. 1.1 Layers of Cloud Services, IaaS, PaaS, SaaS & FaaS.

To take care for instance security constraints cloud consumer must considering network access hardening of instance on public facing interfaces using various could network resources like security group, NACLs etc. In context of high availability consumer must take care for high availability of service by using HA

resources like Load Balancers, Name services etc... As cloud user does not need to create VM / instance in Cloud, i.e. server for performing the intended functionality. Serverless is also popular term for FaaS – Function as a Service. As a Cloud FaaS user just need to write processing logic for function in available coding language supported by FaaS. Implemented Function with processing logic, FaaS is ready to serve the purpose. Lifecycle for the function is until the execution of the FaaS. It does not maintain state of the execution on its own. Unlike traditional instance in Cloud, FaaS is stateless and does not maintain its state on its own. In order to maintain state FaaS can use supporting cloud services like object storage, no-sequel DB, DBs, parameter store etc. FaaS is well suited for computing context of lightweight computing, stateless / light state computing is required. This goes hand in hand with Micro service architecture and IOT platform. FaaS – Function as a Service is available in Public Hosted Cloud such as AWS, Microsoft Azure, Google Cloud, IBM Cloud etc. and Private Hosted Cloud Implementation such as OpenStack Cloud Implementation.

Literature Survey

Continuous Computing Service evolution to various stages like hardware to containers, Function as a Service brings cost effective solutions. There are various Serverless-computing platforms which can be analysis based on multi-level features to evaluate best Serverless platforms. [1] Serverless computing brings abundant amount usage aspects such as resource abstractions, distributed computing, granularity, and application development convenience. [2] Serverless abstraction brings in focus in IT industry. AWS Lambda Implementation of FaaS in AWS cloud have various pros and cons such as cold start, parallel execution. Using various techniques and Lambda feature robust system can be build. [3] Public Cloud Implementation of Serverless such as AWS Lambda, Google Function etc. leads to vendor locking in terms of its usability. Open Source Implementation of Serverless in managed cloud can put some standardization of Implementation of the Serverless service. [4] Serverless service implementation relax developer to focus on Function Code Implementation rather than caring about system Implementation, redundancy, security, abstraction etc. various aspects. [5] Serverless computing provides abstraction such as memory abstraction, networking bandwidth requirement, processing power, etc. Run Time Processing power and memory requirement is configurable in the Serverless computing. Network speed allocation is not having granular control in Serverless computing; there is more opportunity to improve network bandwidth performance for Serverless service. Which is require to address for Network bandwidth intensive application running on Serverless service. [6] Open source framework can be utilize to implement event driver, data processing Serverless service in private cloud. This can benefit various goods in various framework in Open Source for Serverless Implementation. [7] Serverless service can be utilize to build self-content, selfmanaged, self-sizing independent service such as selfcontained website. [8]

Proposed Methodology

A. Architecture

FaaS execution lifecycle is as depicted as in below callflow diagram. FaaS can invoked by internally or External means. It implies when FaaS configured to trigger based on log or some resource/DB events it referred as Internal FaaS Invocation and FaaS can invoked by means of HTTP/HTTPS call this referred as external FaaS call. in consistent state resources like no-sequel DB, DB or parameter store etc. FaaS function instances can run simultaneously based on configured execution rate configured for FaaS. FaaS implemented in various cloud providers range from public cloud providers, private cloud providers and hybrid cloud providers. Variety of options are available, to implement FaaS as service in Cloud. In Open Stack A private cloud Implementation offers liberty of Implementing Services in Cloud as per required technologies such as FaaS can be implemented as standalone Container Service or Container Service managed by Kubernetes engine or by implementing as Virtual Machin Invocation using orchestration module or standalone API Invocation also bare metal VM creation for specialized FaaS requirement for specialize hardware. In Order to Evaluate best suited Implementation technique for FaaS as service in OpenStack various methodologies has been Implemented on OpenStack Rocky release version, Implementation method has been evaluated based on various criteria such as Execution performance, FaaS Invocation time, Implementation complexity, Support for any specialize hardware requirement. For Implementation of FaaS in Open Stack Private Cloud requires Open stack Rocky Release cloud version installed on the Hardware platform with bare minimum services installed. For Implementation of FaaS using bare metal Iron Function Ironic Cloud service needs installed and configured in Private Cloud service.

Implementation Details

Implementation of Open Stack Cloud Rocky Release on Centos 7 (Release: 1810)

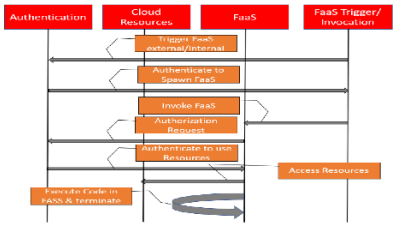

Fig. 3.1 Layers of Cloud Services, IaaS, PaaS, SaaS & FaaS.

When FaaS is invoked it first the request is authenticated via authentication mechanism, only when Invoker is authorized to Invoke FaaS, FaaS is trigger. Based on FaaS logic implemented it can access other cloud resources. Before accessing cloud resources by FaaS function, request authenticated by authentication mechanism to access cloud resources. Post FaaS invoked for execution with code logic provided while deploying the FaaS function. After function execution completes, function instance terminated. For next subsequent request / invocation, same call flow executed. Thus, in this way no state is maintain by FaaS function across function call. State of the FaaS function can maintained by external means by storing the state during the execution phase Controller node having configuration as below:

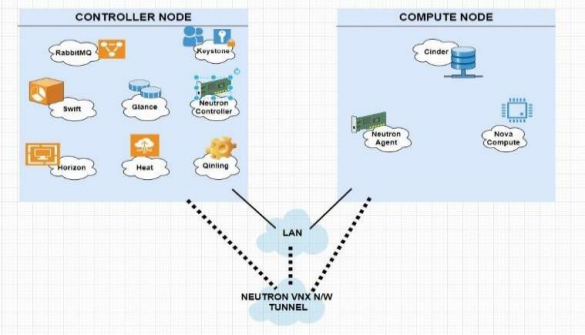

Fig. 3.2 Implementation of Open Stack Cloud with FaaS

- 4 GB RAM
- 4 core Processor
- 100 GB Hard Drive
- Centos7

Compute node-having configuration as below:

- 16 GB RAM
- 12 core Processor
- 100 GB x 4 Hard Drive
- Centos7

Result and Discussions

Considering various aspects of using FaaS is very beneficial to its end users in terms of cost effectiveness, distributed micro service design simplified architecture, securing resource access etc. as described below.

A. Cost effective

Traditional approach to provide implementation to API require Virtual Instance created, configured & deployed with its

Below Open-stack Services are installed on servers for Open-Stack Cloud Implementation with Serverless i.e. FaaS.

- RabbitMQ (Queuing Service)
- Keystone (Identity service)
- Glance (Image Service)
- Nova Compute(Compute Service)
- Neutron (Networking Service)
- Cinder (Block Storage Service)
- Horizon (Dashboard)
- Swift (Object Storage / bucket Service)
- Heat (Orchestration Service)
- Qinglin (Serverless - FaaS Service)

API implementations. The deployment strategy must have consider scalability and highly availability of the computing resource. For public cloud, this leads much more cost even if it is micro service API implementation. Using FaaS as strategy for implementing micro, distributed API is very simplified. User just pay for the time duration during which the FaaS was in execution state.

B. Micro service Distributed Architecture

FaaS Is well suited for micro service based Distributed Architecture; servers do not need to care about complexity of Distributed architecture. FaaS keeps the implementation overhead of application transparent; this makes application deployment simple. This is best scenario to consider for Micro service Distributed architecture.

C. Scalability & Availability

User gets rid of the scalability and availability of the application. FaaS itself is highly auto scalable and available in nature. This makes System Design simplified. FaaS can scales up and scale down automatically as per required load on the API; it can scales from few request per day to thousands of requests per second. FaaS is highly available in nature, even if failure of one of the occurrences another occurrence is ready to serve the request within fraction amount of time. This enables applicable highly scalable & highly available at very low cost. If multiple functions / micro services runs on the same host, application performance degrades due to disk and memory allocation, which indirectly hit network performance. Implementing micro service distributed application using FaaS will significantly increase overall performance of the application, as there will be disk and memory allocation from different underneath hosts of Docker / virtualization layer.

D. Security

As for end user FaaS function is visible only as API it isolates cloud internal resources from end users, thus make is more secure as the backend servers are not visible outside the FaaS function. There is controlled way of implementation of service by limited provisioning of memory and computing cores allocation.

Various Use cases for FaaS

A. Event streaming

FaaS can be trigger from various events generated like system logs, data event, scalability events etc. without

requirement of complex clusters. Event streaming pipeline, queue can empower analysis system; no-sequel DB, object storage can be used for inputs to monitor systems.

B. Image and Video Manipulation

FaaS is well suited for basic Image and video processing which does not require state to retain for subsequent calls. Basic Image and Video operations can performed such as resizing, transformation, cropping, Image to text conversion, creating thumbnails etc.

C. Multi-language Applications

While building Micro service based distributed application, which can leverage multiple cutting-edge programming languages simultaneously for different modules, this makes application development very quick and robust

D. Continuous Integration and Continuous Deployment

FaaS allows Distributed application maintenance very quick and easy. Rather than pushing monolithic update for application, small modules can upgraded rather than waiting for complete monolithic push. This makes application very robust as bugs fixes can easily pushed in CICD environment, as soon as code tested next moment the code is available in production environment.

Conclusions

FaaS is stateless by its nature; state can maintained across FaaS function calls by using supporting cloud resources. This well suited for Micro service based distributed application. There are various implementations of FaaS service in private and public cloud like AWS, Google Cloud, Azure, and OpenStack. Faas implementation is more flexible in Private cloud implementations, as got freedom to implement the FaaS service as per requirement in terms of underneath hardware and supporting execution environment. Using FaaS in public cloud is very cost effective as user pays only for the execution time duration of FaaS. User no need to care about availability and scalability of the application. Micro service designed distributed application in FaaS is highly scalable & provides high throughput. One aspect of using FaaS is to isolate underneath cloud resources isolated from end users / other architectural components.

Thus, FaaS is cutting-edge technology for developing application with multiple languages and quick and robust deployment by its Auto scalable and Highly Available nature. Implementing FaaS in Open-Stack Cloud provide flexibility in terms of runtime support environment like Java, Python, Shell, etc. Also in terms of virtualization framework like container, bare metal service, virtual machine, etc.

References

[1]. Theo Lynn, Pierangelo Rosati, Arnaud Lejeune, Vincent Emeakaroha 'A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms' 2017 IEEE International Conference on Cloud Computing Technologyand Science (CloudCom).

[2]. Erwin van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uta, Alexandru Iosup 'Serverless Is More: From PaaS to Present Cloud Computing' IEEE Internet Computing (Volume: 22, Issue: 5, Sep./Oct. 2018).

[3]. M Daniel Bardsley, Larry Ryan, John Howard 'Serverless Performance and Optimization Strategies' 2018 IEEE International Conference on Smart Cloud.

[4]. Sunil Kumar Mohanty, Gopika Premsankar, Mario Di Francesco 'An evaluation of open source serverless computing frameworks' 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).

[5]. Jose Luis Vazquez-Poletti, Ignacio Mart ln Llorente, 'Serverless Computing: From Planet Mars to the Cloud' Computing in Science & Engineering (Volume: 20, Issue: 6, Nov.-Dec. 1, 2018)

[6]. Jeongchul Kim, Jungae Park, Kyungyong Lee 'Network Resource Isolation in Serverless Cloud Function Service' 2019 IEEE 4th International Workshops on Foundations and Applications of Self Systems (FAS*W).

[7]. Alfonso Perez, Sebasti an Risco, Diana Marıa Naranjo, Miguel Caballer, German Molto 'On-premises Serverless Computing for Event-Driven Data Processing Applications' 2019 IEEE 12th International Conference on Cloud Computing (CLOUD) 7.

[8]. Markus Ast, Martin Gaedke 'Self-containedWeb Components through Serverless Computing' Proceedings of the second International Workshop on Serverless Computing Pages 28-33.