International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Continuous Resilience Testing on Cloud Infrastructure

Nishanth D Aluhonnu[1], Sujata Priyambada Mishra[2]

[1] *Nishanth D Aluhonnu, Dept. of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, Karnataka, India*

[2] *Sujata Priyambada Mishra, Assistant Professor, Dept. of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, Karnataka, India*

Abstract - *Cloud undeniably offers substantial capacity as a way to cut costs, growth agility and reduce risk. But failure to properly plan for and put into effect resiliency can result in an unintentional boom in overall employer risk. Business today rely heavily on cloud computing, this brings immense popularity and difficulties. One of the major problems is fault tolerance: that is breakage due to failures (either due to physical accidents or natural disasters or firewall hacks) may cause heavy revenue shortages. Many users of cloud solutions expect the life of a level of non-stop availability that is now not necessarily engineered into all cloud services. Resiliency testing is done by intentionally causing failures in the cloud system. Resilience of the machine may be defined as its capability to hold functioning even if some components of the system are failing. In a cloud-based infrastructure availability, fault tolerance and resiliency are the most vital advantage. Immense measures ought to be taken so that those services live up and running. This paper introduces 2 such high availability tools called Nomad and Docker-swarm and tests a important business system for resiliency. The importance of resiliency and its key advantages are discussed in this paper.*

Key Words*: *Cloud Computing, Chaos Testing, Automation, Resilience, Fault Tolerant Architecture, Chaos Monkey, Zero Downtime, High Availability.

1.INTRODUCTION

Cloud computing is the processing and garage of records executed with the aid of a remote server. This has emerged as a new enterprise fashionable of the manner applications work and how facts are stored. Most companies, today, are undergoing a cloud local transformation for their services and products to absolutely utilize the characteristics and the advantages at the cloud. Around us although every agency does communicate about having a proprietary cloud or operate on a cloud, some aren't absolutely utilizing the completeness of the cloud.

Chaos Engineering is the discipline of experimenting on a machine in order to construct self- belief in the device's functionality to face up to turbulent conditions in production. The great defence towards sudden failures is to build resilient services. The cloud is all redundancy and fault-tolerance. Since no single factor can assure 100% uptime (or even the priciest hardware ultimately fails), we ought to design a cloud architecture where man or woman additives can fail without affecting the readiness of the cloud system. We ought to be stronger than our weakest link.

1.1 Literature survey

In [1], it discusses present day software-based totally services which are implemented as dispensed systems with complex behaviour and failure modes. Many massive tech organizations are the use of experimentation to verify the reliability of such systems. Chaos engineering refers to this approach, and the underlying principles and the way to use it to run experiments have been mentioned. Start with the aid of defining 'constant country' as some measurable output of a device that indicates regular behaviour. Hypothesize that this consistent kingdom will maintain in each the control organization and the experimental group. Introduce variables that reflect actual world occasions like servers that crash, tough drives that malfunction, network connections that are severed, etc. Try to disprove the hypothesis by way of looking for a difference in constant country between the manage group and the experimental organization.

In [2], an error detection and correction strategy are proposed which became able to detecting and correcting errors in non-linear systems. A system failure often reasons a change inside the critical device parameters and those parameters have been continuously monitored. The operation of the gadget turned into simulated and periodically as compared with the actual version. If there's any distinction between the simulated version and the real version then it method that an error has passed off in the system and changes will must be made. Then the important parameters that are exclusive could be analysed.

In [3], a statistical method became used in fixing problems. The records stream changed into given as enter randomly. The information stream became modelled the use of Non-Homogeneous Poisson's system. The system calculated the range of occurrences of the occasion given the average range of activities that occur at some point of a duration of time. It also described situation which brought about fault reintroduction. Fault re-advent is typically precipitated due to unsuitable debugging. The random facts stream was modelled using Non homogeneous Poisson Process the then system learning become used to characterize the data stream. Clustering primarily based approach became used to in fault detection for the very last stage of blunders detection.

In [4], this paper delivered essential trends in ICT nowadays: the backend of online packages and offerings are transferring to the cloud, and for delay-touchy ones the cloud is being prolonged with fogs. In this paper the mild modification of OpenStack, the mainstream VIM today, which enables it to manipulate a allotted cloud-fog infrastructure is shown. While their answer alleviates the need for strolling OpenStack controllers inside the light-weight edge, it takes into account network aspects which are extremely important in a aid setup with remote fogs. It proposed and analysed an online useful resource orchestration algorithm.

In [5], this paper describes the scope of cloud computing, the current studies within the area. Discusses vital concept of virtualization, development of cloud hardware and software, the before and after of virtualization, its position in cloud computing, brief view approximately hypervisor, garage virtualization, server virtualization and advantages of virtualization.

2. RESILIENT ARCHITECTURE

Particular features of reactive architecture.

- **Elasticity** - This shows that we can expand or scale the application to the needs as specified. This also helps in the system to be resilient.
- **Responsive** - This ensures that the system is alert for aware of the happenings in the system. It ensures that the system always knows what's going on around it and help system to be in its desired state.
- **Resilient**- This is to ensure that the system is always up and running that is the robustness of the system is very high and it ensures that the system is always available to whatever the user requests and there is no denial of service.

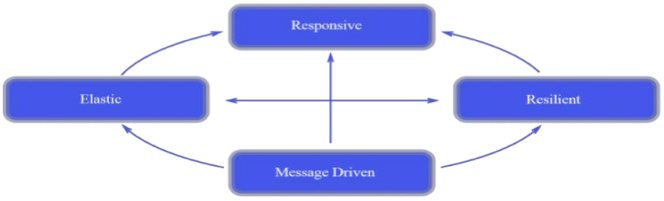

Fig -1: Reactive Architecture Characteristics

3. RESULTS AND DISCUSSION

To test the resiliency of the assurance system several specific process/container/ethernet disrupting tools were injected into the system. The behaviour of each component was closely analysed. Each component behaved differently for different tool injections.
Discussions of two such tools have been made below:

3.1 Specific integral process kill

Killing a linux process was achieved with a python script running OS commands and executing the kill -9 command after achieving the PID (process ID) from the process list that is obtained from ps -ef | grep <process name> command. This python script was used to automatically nuke integral tools of the Service assurance architecture and reactions of each tool in the assurance architecture was noted.

Signal #	Name	Description
1	SIGHUP	Hang up signal. Programs can listen for this signal and act upon it. This signal is sent to processes running in a terminal when you close the terminal.
2	SIGINT	Interrupt signal. This signal is given to processes to interrupt them. Programs can process this signal and act upon it. You can also issue this signal directly by typing Ctrl-c in the terminal window where the program is running.
15	SIGTERM	Termination signal. This signal is given to processes to terminate them. Again, programs can process this signal and act upon it. This is the default signal sent by the `kill` command if no signal is specified.
9	SIGKILL	Kill signal. This signal causes the immediate termination of the process by the Linux kernel. Programs cannot listen for this signal.

Fig -2: List of signals the kill command supports

3.2 High availability of containers with Docker swarm

Tools like **docker-swarm** and Hashicorp's **Nomad** to make a system highly available. This is achieved by making replicas of different tools on all available systems/nodes in the network. Autonomously creating new replicas when new nodes are added to the network and scaling up or down based on usage. This has lengthy been the case that to build structures with this type of reliability and availability means big costs for companies.

```
[root@manager vagrant]# docker service ls
ID            NAME           MODE         REPLICAS  IMAGE
1xi5ys7sz5jw  vagrant_db     replicated   5/5       vagrant_db
z13l4ztyw3rs  vagrant_flask  replicated   5/5       vagrant_flask
[root@manager vagrant]# docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
1i1r2e8e4tz83fkd0ycdb4pli *   manager   Ready   Active        Leader
x2f2h48l0wuvsgrhebv1ekyap     worker1   Down    Active
x9o2tgolgzrfzaf6rqimoef6w     worker2   Down    Active
[root@manager vagrant]#
```

Fig -3: Worker and Manager nodes setup using docker swarm

For something like this, it is not enough to certainly have a blackout of cluster of servers in a datacenter, there need to additionally have more than one redundant strength sources for the datacenter and even to have replication between more than one geographical location in case of disasters. With the exception of very massive,

multinational companies, nearly no-one should afford the sort of setup. With the appearance of IaaS and PaaS providers, also, the costs of building the sort of carrier have reduced dramatically. This may be accomplished with tools like Nomad and docker swarm.

```
[root@manager .bin]# ./pumba kill -l 4 re2:vagrant_d
dcoker [root@manager .bin]# docker service ls
ID            NAME           MODE        REPLICAS  IMAGE
1xi5ys7sz5jw  vagrant_db     replicated  1/5       vagrant_db
z13l4ztyw3rs  vagrant_flask  replicated  5/5       vagrant_flask
[root@manager .bin]# docker service ls
ID            NAME           MODE        REPLICAS  IMAGE
1xi5ys7sz5jw  vagrant_db     replicated  5/5       vagrant_db
z13l4ztyw3rs  vagrant_flask  replicated  5/5       vagrant_flask
[root@manager .bin]# ./pumba kill -l 4 re2:vagrant_d
[root@manager .bin]# docker service ls
ID            NAME           MODE        REPLICAS  IMAGE
1xi5ys7sz5jw  vagrant_db     replicated  3/5       vagrant_db
z13l4ztyw3rs  vagrant_flask  replicated  5/5       vagrant_flask
[root@manager .bin]#
```

Fig -4: Replicas that are being killed are automatically respawned

4. DISCUSSION ON NOMAD

Nomad is a modular workload orchestrator that enables enterprise to quickly complete deployment and direct the use of a single, cohesive workflow to any simple or containerized application. Nomad can run a number of applications including Docker, non-containerized, microservice, and batch. Nomad allows developers to use the declarative as-code infrastructure to deploy applications. Nomad uses bin packing to efficiently employ time table jobs and optimize the use of resources. Nomad is compatible with macOS, Windows and Linux. Nomad is broadly adopted and used in production by using SAP, Roblox, eBay, Trivago, PagerDuty, Pandora, Target, Citadel.

Key Advantages :

• **To Deploy Docker Containers and Basic Applications:** Nomad is flexible as an orchestrator which allows companies to run docker containers, basic and batch scripts collectively on identical architecture.
• **Simple & Reliable:** Nomad autonomously controls apps, hosts and motive force disruptions. Nomad is a distributed and robust program, the use of electing leaders and mirroring nation to have excessive availability in the event of failures.
• **GPU Support and Device Plugins:** Nomad gives built in guide for GPU usage such as gadget learning (ML) and artificial intelligence (AI).
• **Multi-Region:** Nomad has Multi-Area Company Native Guide. This built in tool allows for the collective connection of multiple clusters, which in effect allows builders to set up work to any nodes in any region.
• **Proven Scalability:** Nomad is associated with a little bit of luck, which increases throughput and reduces latency for workloads. Nomad was developed in real-world production environments to scale up to clusters of 10K+ nodes.

5. DISCUSSION ON DOCKER SWARM

A swarm is made up of more than one Docker hosts running in swarm mode and serving as managers (to monitor membership and delegation) and staff (who run swarm bids). A given host of the Docker can be a manager, an employee, or perform any role. When you build a provider, you identify its most successful state (variety of replicas, accessible community and storage resources, ports that the carrier exposes to the world outside of the gate, and more). Docker works to preserve the ideal condition. For example, if an employee node is unavailable, Docker schedules obligations of that node on different nodes. A task is a walking box that is part of a swarm provider and operated as opposed to a separate box via a swarm manager [8].
Some of the primary benefits of swarm systems over standalone containers is that you can change the configuration of a carrier, including the networks and volumes to which it is far connected, without the need to reboot the device manually. Docker will uninstall the configuration, prevent carrier duties with the outdated configuration and build new ones that suit the configuration you want.
When Docker goes for swarm mode walks, you can always run standalone containers on some of the swarm-participating Docker hosts, as well as swarm services. A main distinction between stand-alone packing containers and swarm offers is that only swarm managers can control a swarm when starting stand-alone bins on any daemon. Docker daemons can take part in a swarm as managers, workers, or both.

6. CONCLUSIONS

High-availability is the cloud's holy grail. It represents the concept of having access to infrastructure, facilities and records anywhere and anywhere, and is the enabler of dreams of a future with companies and not using physical offices or multinational businesses with fully integrated and centralized IT systems. The quality also has to do with reliability: a service that is on 24x7 but goes offline continuously is useless. To order for a carrier to have really high- availability, it is not best to be always-on, but to have multiple reliability "nines" (99.999 ...) to addition.
Some of the primary advantages of swarm services over individual containers is that it can change the configuration of a service, along with the networks and volumes to which it is connected, without having to restart the service manually. Docker must remove the configuration, forestall the carrier duties with the outdated configuration, and build new ones that suit the desired system.

REFERENCES

[1] DAli Basiri , Niosha Behnam , Ruud de Rooij, et al, "Chaos Engineering", 2016 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE), Huhhot, 2016, pp. 506-510.

[2] M. Polycarpou and A. J. Helmicki, "Automated fault detection and accommodation: a learning systems approach," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 25, no. 11, pp. 1447-1458, Nov. 1995.

[3] Hoang Pham, L. Nordmann and Zuemei Zhang, "A general imperfect-software debugging model with S-shaped fault-detection rate," in IEEE Transactions on Reliability, vol. 48, no. 2, pp. 169-175, June 1999.

[4] David Haja, Marton Szabo, Mark Szalay, et al, "How to orchestrate a distributed OpenStack", 2018 IEEE Conference on Computer Communications, Bremen, 2018, pp. 1-8.

[5] Nancy Jain, Sakshi Choudary, "Overview of Virtualization in Cloud Computing" 2016 Symposium on Colossal Data Analysis and Networking (CDAN), India, CO, 2016, pp. 1-5.

[6] Kazuhira Okumoto, Abhaya Asthana, Rashid Mijumbi, "BRACE: Cloud-based Software Reliability Assurance" 2017 (ISSRE), Yogyakarta, 2017, pp. 154-157.

[7] Jurgin Cito, Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research , Springer 2017.

[8] Chia Chen Chang, A Kubernetes Based Monitoring Platform for Dynamic Cloud Resource Provisioning , IEEE 2017.

[9] Minxin Du et al., “Privacy-Preserving Indexing and Query Processing for Secure Dynamic Cloud Storage” .Vol 13, pp. 2320-2332. IEEE Transactions, 2018.

[10] Bahram Hajimirzaei et al.,. “Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm”. Elsevier, January 2018.

[11] Hong Zhong et al., “Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage”. Springer, 2016.

[12] Colin Boyd et al., “Security Notions for Cloud Storage and Deduplication". Springer, 2016.

[13] Dietz, Marietheres, and Günther Pernul, Big Log Data Stream Processing: Adapting an Anomaly Detection Technique International Conference on Database and Expert Systems Applications. Springer, Cham ,2018.

[14] Miranskyy, Andriy, et al, Operational-log analysis for big data systems: Challenges and solutions, IEEE Software 33.2 (2016): 52-59.

[15] Armin Balalie,et al, Microservices architecture enabling DevOps, IEEE 2016.