

A CloudFormation template using AWS
golden standards to avoid cascading failures
in hardware

MSc in Cloud Computing
Research Project

Arnauv Kaushik
x19231661

School of Computing
National College of Ireland

Supervisor: Jitender Kumar Sharma

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Arnav Kaushik
Student ID:	x19231661
Programme:	MSc Cloud Computing
Year:	September 2020
Module:	MSc Research Project
Supervisor:	Jitender Sharma
Submission Due Date:	16th August 2021
Project Title:	A CloudFormation template using AWS golden standards to avoid cascading failures in hardware
Word Count:	6893
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	ARNAUV KAUSHIK
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A CloudFormation template using AWS golden standards to avoid cascading failures in hardware

Arnaud Kaushik
x19231661

Abstract

The world of IT has been revolutionized by introduction of cloud computing, reason being that one does not need to invest enormous amounts of money to setup a business today, the resources can be made readily available instead of waiting for long, the biggest advantage of cloud impacts every commoner as well, the data can be stored to cloud and can be viewed anywhere, this directly results in remote access and data recovery. Apart from the cloud itself, the sub-technologies or the applications of cloud computing such as AI and ML are taking on the industry today by storm. However, with time and increase in the competition, getting these on-demand IT services is no longer a stand-out feature, customers and client today focus on non-functional requirements as well such as availability, reliability, scalability etc. The main reason can be linked to user-experience as well and can make or break a deal. In a quest to expand worldwide and to provide for a large number of present and future clients, providers and expanding their datacenters in terms of numbers. This results in them having large number of complex hardware that need to communicate with one another almost all the time even when not being used actively, this may result in cascading failures in case a core element fails or shuts down abruptly. Currently, there is no single paper that covers all of the main non-functional requirements combined with modern software practices such as microservices, queues so as to use software as a shield to defend against any possible failures arising due to hardware. This paper aims to use software best practices to make a cloud-environment free of disruptions and hence providing the much-needed user-experience that stands out in today's market.

1 Introduction

We realize that cloud services are the consequence of suppliers having CDCs(Cloud Data Centers) spread across a topographical region and giving end clients the IT-related services with a pay-as-you-go model. There is nothing unexpected that these datacenters in reality comprise of exceptionally interconnected frameworks which rely upon one another. Also, in the greater part of the cases they are limited by interconnections and network and communications. These interconnected frameworks clear a path for a worry in regard to negative results if a significant segment stops abruptly because of whatever reason. This worry merits featuring on the grounds that in specific cases, a falling arrangement of disappointments may amount to an enormous disturbance of services which might be basic in nature. This is the place where this paper comes in, this paper attempts to answer whether we can abstain from cascading failures in hardware by utilizing programming

and design best practices or not. Cascading failures here allude to a cascading type of influence or a chain reaction, wherein an occasion triggers a progression of events, for this situation that is similar to one piece of an application coming up short and bringing about failure of the application all in all. The expression 'using software best practices' is used to demonstrate a connection between the hardware and software, programming impacts equipment as in for a client composing a code to cooperate with cloud benefits really calls the devoted APIs and henceforth, we can write code to act against certain backend failures, using auto scaling to consistently keep an instance running is only one of numerous potential ideas. Going to the implementation part, this paper admires the best programming designs used to keep application segments promptly accessible just as inclined to any kind of possible failures. At long last, services, for example, CloudFormation are viewed as a way to arrange, oversee and modify code, we can without a doubt influence hardware as far as progressing, repeating it. This paper targets utilizing a golden standard of practices in the event of a disaster, however, to utilize a similar in advance to keep away from any opportunity of something similar. The essential inspiration to chip away at this point is that dependability goes hand in hand with QoS or the Quality of Service which helps in deciding a cloud provider standing among cloud suppliers. This paper endeavors to and acknowledges the demand to differentiate different cloud service models to measure contrasts and see the extent of including some other great practices to upgrade the model further. The principal inspiration for utilizing the strategies, procedures and practices came out as an after-effect of a procedure to admire the lead provider in cloud and cloud services. AWS(Amazon Web Services) isn't just at the top but manifolds ahead of other contenders consolidated. Looking for an optimal engineering for programming projects, this paper looks into specific standards and focuses referenced by Amazon itself, puts them to test for their legitimacy, examination with next-in-line contender and looking for any inadequacies. This paper focuses on the end clients, the ones who utilize the end result cloud services without any respects to the backend working at all, moreover, it is focused on for experts and understudies the same who manage cloud benefits and related applications to any degree, the eventual outcomes can be observed by non-specialized people too who are recently familiar to cloud or any of its sub-area. Concerning esteem in the related innovation area, considering every referred to paper and accompanying a brought together form of every best practice and tips makes this paper worth a read as it dispenses with the need to examine further as far as another and imaginative way to deal with accomplish high accessibility and accordingly unwavering quality. The base subject itself merits assessing and plunging into on the grounds that client experience goes inseparably with the assistance quality and a cloud's customers may increment or decline according to the verbal exchange dependent on cloud accessibility, the application dependability and henceforth the Help Level arrangement. The approach(es) referenced in this paper are doable enough dependent on the way that they are reasonable by an expert and understudy the same. The essential justification this is that the essential assistance referenced is AWS CloudFormation, which is an allowed to utilize administration independent and has no additional expense to utilize separated from segments produced from it. Besides, cloud suppliers give plentiful assets as far as documentation and restrictive free utilization which cost nothing. A few instances of this incorporate the free documentations present for AWS, GCP and AWS' complementary plan which gives one year of let loose use of specific services to a sensible amount, GCP gives a heavy beginning entirety to investigate its cloud services. The paper is isolated into committed segments which can be portrayed in a word beginning with section2 which is the writing

audit, which covers research that started this paper, let it work as far as investigating the subject, ebb and flow situation, difficulties and shows light to conceivable solution(s) alongside the exploration specialty, which contains the significance of the examination papers. Section 3 is named 'Exploration strategies and details', which talk about the game plan, they display certain designs and reprimand them to assess for existing and missing focuses identifying with accessibility and applications having the option to bounce back from a failed part. We additionally move our concentration towards CloudFormation, a brief intro followed by a brief look at code expected to carry out our designs being referred to. This is trailed by a brief of our testing approach that is trailed by Gantt chart for the examination proposition's execution. At long last, section 4 finishes up the paper while bringing up significant places of thought and looking at the models being referred to head on against every single significant segment assuming it fails.

2 Related Work

In this section we will discuss the previous research relating to cascading failures in hardware and software and cloud reliability. Now a days most of the organisations are adapting cloud services. Some of the basic features provided by cloud such as reliability, scalability etc are key attraction and reason for the rapid adaptation of cloud services. Providing good performance and standards are vitals aspect of the cloud providers, this can be achieved by collective smooth service of back-end and front-end services. The appearance of cascading failures in cloud services can result in a huge scale of interruption and degradation of the quality of service. This disruption might affect and violate SLAs or Service Level Agreements with the organisations. In below section we will see in details of previous research on different topics. In section 2.1 we will review the research with Reliability in cloud computing services, in section 2.2 we will review the previous research with Cascading failures in software and hardware and in section 2.3 we will review previous research to understand why AWS is selected for this research.

2.1 Reliability in Cloud Computing

The cloud service providers have attracted a huge scale of possible organisations with flexibility and many offerings. The pay-as-you-go model, flexible subscription and anytime support services have helped the organisations to handle efficient cost, time and other business challenges. The research paper Buyya et al. (2018), highlights how the features provided by cloud services has attracted possible business, but at the same time it has emerged challenges in terms of scalability, reliability, security, sustainability etc. This paper discusses the existing challenges, as well as future aspects of these with cloud services. The paper presents a manifesto related to issues with present cloud service provider, which can be analyzed for future cloud computing. This shows that improvement in reliability of cloud computing services is very vital aspect. The research paper Tian et al. (2020) proposes a framework by combining the identification of the risk and proactive actions follow up for advancement in efficiency and reliability of cloud computing services. The paper describes the failure in cloud services as inevitable, but understanding these failures such as fault tolerance and their recovery we can improve the cloud service. The proposed framework is first made to understand these failures and integration of failure

risk prediction in cloud services. Further a tree based model is implemented for predicting risks in cloud failure, with integration of cloud failure risk and the tree based model. Further to test the framework a case study from google cluster is considered, with storage of operational data containing volume of 400GB for twenty-nine days consecutively. The research paper Mesbahi et al. (2018), highlights how the reliability and high availability are vital concern for cloud computing service providers. According to this paper these two characteristics are very important to maintain the customer trust and satisfaction towards the cloud service. The paper presents a study which has covered all the aspects of challenges and problem in cloud computing service. This was achieved by analysing four questions that are 'What?', 'Where?', 'How?' and 'When?'.

The paper Alam et al. (2018), proposes an evaluation model for evaluation of reliability in cloud services, using effective manner. As we know reliability is one of the main aspect of cloud service provider for good customer satisfaction. To improve this aspect from cloud computing service provider, the paper proposes an evacuation model for improvement of reliability in cloud computing services. For implementation of the proposed evacuation model, the author understands and highlights the strategies involved in cloud failures. The paper also considers other failures from different type of domain applied in cloud environment for a proper and thorough evaluation of cloud reliability in the services. The paper has implemented the proposed evaluation model by execution of the failures, and the results of this research is communicated using simulation. The paper Li et al. (2020), proposes a creation strategy based on gray Markov chain with dynamic replica. Multiple data replicas can be created from multi replica strategy. This will eventually introduce improvement with the availability of data and service quality of the data. But at the same time the Data Nodes required for this implementation are limited in number as compared to the demand presented by the user. And hence, due to increase in data replication the load on data storage is increased. To resolve this issue the paper has proposed a creation strategy based on gray Markov chain to achieve dynamic replica. As per the paper, whenever there is requirement of new replicas, these replicas are created in DataNodes which impact the load balancing of the cloud service provider. Hence, by keeping this in mind the author has proposed a Fast Non-dominated Sorting Genetic algorithm for strategies involved in the replica placement. In conclusion the proposed experiments have shown effectiveness in handling the challenge of load balancing due to data replica placements.

The paper Cui & Li (2020), proposes a system movement space and mapping theory for system to study the significant movement in process in system and the change in the reliability. The system structure of IoT is enhancing towards complexity, which is impacting reliability in the cloud services and is emerging as concern for the users. The paper describes the reliability, and faulty in process and big data for industries in engineering. The author here presents a theoretical basis of explanation or the methods for the study in IoT reliability. Two methods namely System Movement Space (SMS) and System Mapping theory (SMT) are proposed. As we know for any cloud service cloud data center is critical and hence the characteristics related to cloud data center such as infrastructure, and reliability service is very vital for a proper service delivery. The paper Li et al. (2021), proposes a hierarchical colored generalized stochastic petri net to bridge or reduce the gap between the existing work and reality. As discussed above the cloud data center are very vital for any cloud services hence, it directly impacts the reliability factor. The proposed framework evaluates the reliability service by the sim-

ulation of Monte Carlo. The proposed modeling and simulation are implemented over the cloud data center of insurance company and the strategies applied by them for cost management related to operation, maintenance and configuration of the cloud data center.

The paper Mahmoud et al. (2018), presents an approach for service based development of software application. The increase in bandwidth , reliability and accessibility is enhancing the developers to select the cloud service SaaS, but they face a challenge of version control and management of discoveries in software development. To bridge this gap the proposed approach was built with distributed service. The implementation and result of this approach is described in this paper. An SLA-based approach for achieving reliability in private cloud is proposed in the paper Malatpure et al. (2017). In the implementation of the proposed approach the author examines how the reliability of SLAs are formulated, identification of the customer key and usage of accelerated testing for validating the implementation. For proper validation of the model the author has created a private cloud SaaS. In conclusion the author shows the result of the simulation using the Microsoft Azure development cycle, while describing the analysis over the issue of reliability surfaced in the process of implementation. In any cloud platform we have observed that the infrastructure of Cloud-management has played a significant role as cloud computing stacks. The high availability feature of cloud management is stated as one of the crucial requirement by the paper Liu et al. (2014). But at the same time the infrastructure of cloud management shows high complexity. The paper states that a very small amount of focus was done with quantitative analysis of cloud management availability however, many research has been done for increase in reliability and availability. The paper proposes a method for availability benchmark for a new cloud management infrastructure. This was achieved by implementation of measurement method over cloud customer or provider. The paper Benchara & Youssfi (2021), proposes a method for clustered distribution of HPC models over medical image processing. The paper highlights the challenge such as communication cost, which impacts the scalability of the cloud service. The proposed approach of K-mean method is performed within a team of micro-services under distributed services by HPC. The previous research shows how reliability has impacted the cloud service provider and the research in this field to evaluate the challenge and mitigate the existing issue.

2.2 Cascading failure in software and hardware

Addressing cascading failure in cloud services are very important, as it grows with time and the impact becomes very huge. Cascading failures are the failures which gradually grows with time and in can lead to system failure and enhancing the probability of the system portion failure as well. The research paper Babalola et al. (2016), proposes an adaptive multi-agent algorithm system, whose technique can be implemented over the cascading failures with facing any loss. The paper highlights the nature and complexity of cascading failures in the cloud service providers and how these failure lead to a huge monetary loss in cloud computing services. The proposed approach is compatible with system of various size. The approach uses a combination of mathematical heuristically combined selection of sensitivities. For implementation of this research a real-time cascading issue was faced and resolved using the proposed approach. The issue of cascading failure tends to cause many challenges such as SLA violations, and monetary disruption etc. The research paper Wang et al. (2018), proposes a CFRS that is Cascading Fail-

ure Resilience System which comprises of three methods that are namely, VM backup set placement (VMset), Overload-Avoidance VM Reassignment (OAVR) and Dynamic Oversubscription Ratio Adjustment (DOA). The paper highlights that in any cloud service, due to cascading failure any physical machine is impacted then, the workload gets transferred to to different physical machine so that the process continues. But at the same time the new physical machine which has suddenly received workload due to cascading failure in another machine faces overloading. To resolve this the paper proposed Cascading Failure Resilience System is implemented. The result of the paper shows that the proposed approach has shown good performance with multiple domain failure and workload balance of physical machine due to cascading failures.

The research paper Xing (2020), presents the methodologies for reviewing the cascading failure and the impact over the reliability of cloud services and IoT. As mentioned above cascading failures are the issues whose outbreak is elongated with time, that is if one failure is occurred a second failure is triggered. So to understand the reason and the steps to resolve these cascading failure the paper highlights the reasons and the behavior shown by them. This paper further proposes mitigation solutions for the outbreak of cascading failure.

Similarly, the paper Liu et al. (2012), proposes a protection scheme with a wide area based multi agent system for prevention of voltage instability induced due to cascading failure. The paper highlights that in any cloud service, due to cascading failure any physical machine is impacted then, the workload gets transferred to to different physical machine so that the process continues. But at the same time the new physical machine which has suddenly received workload due to cascading failure in another machine faces overloading. In the proposed approach external relays and controllers work efficiently as per the cascading failure resolution. The result is presented by simulation, which shows that the proposed strategy is very effective and provides a good performance for stability of induced cascading failures. The paper Lan et al. (2010), highlights the evaluation of credibility in software and testing in software. According tot the author this has evolved as a theoretical challenge, and this issue needs to be addressed sooner, because of the complex and large structure of software. The paper discusses the results obtained in some of the experiments, and concludes that the credibility of executing software with less test cases provide a better performance and a critical node guarantee.

2.3 Why AWS

For implementation of this research experiment, we had to select a cloud service provider which would elaborate the challenge and adapt to the market with best practices. So here in this section we will review some research papers which will help us understand and select the best public cloud service provider. The research paper Kamal et al. (2020), highlights how the selection of suitable cloud service provider has emerged as a challenge. This papers presents a comparison between the computational service, storage and infrastructure provided by leading cloud service providers globally. For this the paper has selected three cloud computing service providers that are Microsoft Azure, Amazon Web Services and Google Cloud Services. Different aspects such as services, flexibility, price, advantages and other features are elaborated for all the three cloud computing service provider. By the comparison from all these aspects the paper concludes that AWS was resulted in leading, whereas Microsoft azure has shown good performance in SaaS and

GCP is enhancing its leading service PaaS in Google Cloud platform. Similarly, the research paper van Vliet et al. (2013) highlights the reliance and reliability in AWS services. According to the paper there are various factors impacting the reliability of any cloud provider, but the most vital aspect is resilience. A process of recovery from any disruption occurred in workload due to infrastructure or services is known as Resilience. To achieve this the cloud service provider need to acquire the on demand computer service dynamically and mitigate the present disruption occurred in the service. The research paper Kantaria et al. (n.d.), highlights how and why selection of AWS environment is better than choosing any other cloud service provider such as Azure or GCP. The paper addresses all topics from creation to final implementation with the cloud services. This article focuses on the challenges and benefits of using AWS than other cloud service provider.

Similarly in paper Campbell (n.d.), vital components of automated AWS infrastructure are explained by the author. The author has presented deep analysis in terms of infrastructure and the driving concept of API and also covers the benefits in terms of reliability, agility and life-cycle automation of infrastructure. The paper also describes the how the new architecture presented are reliable and flexible. The author describes regarding the security and reproducibility of these services, with addition of DevOps. The in-depth description of AWS CloudFormation such as Sceptres, toposphere, Terraform etc is also defined. The technological aspects as well as the organisational aspects are very nicely depicted in this paper and explains why the tooling solutions of AWS as services to customer is best in all terms. The paper Wittig & Wittig (2018), acknowledges the currently present and running AWS services and provides us an insight of the steps taken by AWS for resolution of the issue related to storage and its solutions. The author highlights the limitations and challenges in using an on-premise data center, which includes the necessity of resources and maintenance of the resources, cost limitations and challenges. Management of all these challenges in an on-premise data center creates a dilemma for the organisation, and helps us understand the benefits of using AWS services for resolution of all these issues and challenges. A step by step workflow of implementation of AWS service cloudformation is described and how it helps us overcome these challenges. The AWS best practices are explained and analysed across the existing issue of On-premise data center. The author presents an insight of cloudformation and its characteristics of maintaining consistency in implementation of infrastructure in AWS platform. As we know if any issue is presented to the developer they will have their own way of handling and resolving the issue. To improve and omit this discrepancy and confusion in later phase, the author proposes a CloudFormation template. These previous paper has provided us the reference for adoption of AWS to understand and progress in the research. Hence, for implementation of this research we have used AWS as cloud service.

3 Research Methodology

In this section, we will firstly focus on the research question. Then we will direct some possible approaches towards reaching the goal. Because it is important to justify why a particular method is better than others, this same argument has a dedicated subsection as well. In section 3.1, we give a quick recap to the research question before exploring

how to get towards a path to it. Section 3.2 is about the approach, being the basic step and also the initial one, it is very important to have a plan of action and this is what is described herein. After the approach, we see discuss about some alternatives choices applicable for the research paper, this is better described in section 3.3, this particular section also explain why a particular alternative stands out from others and thus leads other aspects in this paper.

3.1 Recap of research question

A quick recap of the research question is that we seek a method to synchronize hardware using software and also impose best-practices recommended. Manipulating hardware through software is a required concept in modern times as both components can be controlled using one of them, in doing so, all attention is focused on a single component and hence this increase of focus will reduce the margin for error by avoiding the need to focus on hardware and software individually and configure both to work in sync.

3.2 Approach

Regarding the approach which directly affects the question is the use of 'infra as a code' methodology which perfectly fits the description and use-case for this research. This methodology enables manipulating hardware components using software. Any code written needs a clarification of what components are required based on the use-case, when this script is run, the relevant components of script call the relevant APIs and at the data-centre level, the resulting hardware components are spun up and/or revised. Public cloud providers as well as several third parties have their personal alternatives to this. A few alternatives widely popular are *Iac Tools list* (n.d.) :

- AWS CloudFormation - As the name suggests, this is a cloud service provided by Amazon Cloud Services. herein we write what are called as 'templates', these templates can provision huge collections of hardware and third-party services as well, all this can be updated, deleted as a single unit. Moreover, this is not limited to a particular user or geographical region, the template can comprise of multiple accounts spread across multiple regions.
- Terraform - One of the offerings from Hashicorp, Terraform is an open source tool that primarily uses a Command-line Interface (CLI) to enter lines of code and then use to work on APIs of many cloud services. Its main advantage is that it is not at all specific to any particular cloud provider.
- Google Cloud Deployment Manager - An offering of the Google Cloud Platform, this services enables us to automate creation of stacks that have Google's cloud services as its components, using a single template, we can make various services work together.
- Azure Resource Manager - Azure from Microsoft provided this offering under Azure DevOps so as to provision and maintain various components under one single roof. Numerous components can be handled simultaneously by mentioning the type and relevant parameters after their provision.

It is visually evident that all of these services and tools can get the job done. In the very next section, we see what to pick over others and why.

3.3 Alternative choices

It is very important to select the best possible choice by comparing the available options at hand. Referring to sources such as [Misc Ref 2] :

- Regarding the choice, AWS Cloudformation is a clear choice. Not only because AWS being the clear cloud provider to be considered, but also because CloudFormation is perfect in case of AWS-specific services and some third-party add-ons. Terraform from Hashicorp is the second best competitor because of its flexibility, it removes the cloud-specific restriction. Still, AWS CloudFormation has an upper hand because it can be used single-handedly to work on more than 350+ services [Misc ref 2], that too from a single point of origin, a single template defining everything needed.

In this research paper, the advantage of using CloudFormation is enormous and unparalleled to any other alternative. The initial research follows AWS-defined standards to achieve an application made and deployed with best practices in mind. Being in the AWS environment for the entire procedure, right from start with only help the cause of this paper.

Choice of method, its limitations.

The method chosen is to consider the mostly used AWS services and combine them in a single template, this way, a single template can be used by most individuals, companies to get a head-start in setting up their application environments. The only limitation clear is that as the number of services increases, so does the complexity and the responsibility to make them all work together and adjust the parameters accordingly.

Advantage in the market

- The official AWS website may have some initial basic templates for users to try, but there is a need to have enterprise-level or templates. to be available for individuals and companies to make use of.
- These templates can be uploaded at a dedicated site to be bought or can be used to sell at AWS marketplace and other related places.

Advantages of IaaS

- The key advantage here is that developers won't need to focus on setting up the infrastructure, they can focus on software components better.
- For those involved in infra services, having infra-as-a-code saves them creating everything from scratch, they will just be needing to configure how the components will behave and interact.
- This will in general be a boost to initial stages on the software development lifecycle.
- Any changes required are done on a single template file. There is no need to chase after individual components and worry what's wrong like in a monolithic application.

Other Aspects AWS Well-architected Framework

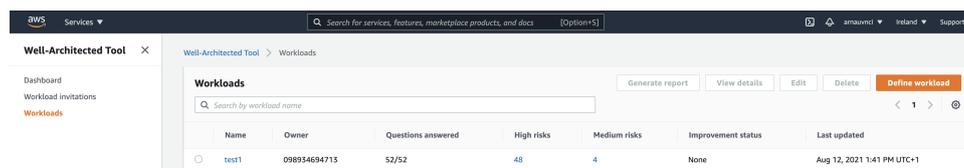
The main idea is to make any existing application ready to be up-to date with modern tools and technology, keeping in mind any possible issues and be ready beforehand with a relevant solution/work.

This template can be seen as one that lays down an application environment for the application. Then, a mere configuration can get things up and running.

AWS Well-architected Tool is a tool provided by AWS that enables anyone to line-up better with the AWS Well-architected framework. The tool lets anyone input their current workload and matches them against what is the AWS golden-standard. There are five main pillars which contribute together to form an ideal application workload, there as follows :

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization

Each pillar has a dedicated set of questions to be answered, most of the questions have four to five options, based on the choices made, the tool gives a report status of what pillars are standing at risk and thus need to be worked upon. Pillars needing immediate attention are categorised as 'High Risk'. Figure 1 shows a short and simple summary of a demo workload entered. It accurately shows how many questions answered were away from the best practice advised. The total question count is 52, spread almost equally across all five pillars mentioned above.



The screenshot shows the AWS Well-Architected Tool interface. The main content area displays a table of workloads. The table has columns for Name, Owner, Questions answered, High risks, Medium risks, Improvement status, and Last updated. A single workload named 'test1' is listed with 52 questions answered, 48 high risks, and 4 medium risks. The improvement status is 'None' and it was last updated on Aug 12, 2021 at 1:41 PM UTC-1.

Name	Owner	Questions answered	High risks	Medium risks	Improvement status	Last updated
test1	098934694713	52/52	48	4	None	Aug 12, 2021 1:41 PM UTC-1

Figure 1: AWS Well-architected tool

4 Design Specifications

This section tells about the main design involved. The design goes hand-in-hand with the main idea. As it is clear by now, the implementation revolves around an AWS cloud setup and that too having the CloudFormation template at its heart. The design specifications have the CloudFormation service containing a stack template. Other than this, we have a dedicated S3 bucket that is used to store templates made. The initial design has only these two components and this is what basically takes to build any scale of hardware stack, this is the power of CloudFormation. After a successful implementation and thus after the template runs, there can be any number of individual and duplicate cloud components, all that totally depends on the template chosen/created by the user.

4.1 Why AWS?

”Why AWS” is a fair and valid question. There is a straightforward answer to this:

AWS has a bigger market reach and more people can relate to cloud services/technologies via AWS. AWS is the clear winner in terms of working standards, principles and even best-practices as discussed in the related work. A wider audience can be reached when talking in terms of AWS technologies compared to other cloud-providers.

The main components include CloudFormation, obviously, the research revolves around this service, this is a key component as this 'infra as a code' service is what acts as a link between hardware and software. S3, The Simple Storage Service can be used for a variety of use-cases. S3 can be used to store CloudFormation templates, can be used to store application code, moreover, can be used as a code-versioning mechanism as well. One of the most common use-case is using S3 for storage and backup. S3 is a core component to make any application multi-AZ and hence more available and reliable.

5 Implementation

The implementation part is not limited to running a CloudFormation template, setting up the components is a key requirement for success. The configuration manual associated with this paper is the go-to regarding the implementation part. The easier path would have been to use any existing CloudFormation template and then use it by modifying certain parameters. This would not have been helpful as the main idea is to include as many as cloud services as possible to make developers and infrastructure team free from any sort of provisioning apart from basic parameter and credentials setting.

The core steps include :

- logging in to the AWS Management Console.
- Creating a stack template from scratch from CloudFormation service page.
- Running the template.
- Filling in the required parameters as required for individual components.

6 Evaluation

Aim:

The main aim is to make any software environment reliable by making it safe from any issues that may be an obstacle to a good user experience. Complex datacenters having tightly coupled components are highly prone to cascading failures if something bad were to happen.

State-of-the-art :

Infra as a Code(IaaC) or Infra as Code(IaC) is not a standard practice. CloudFormation is considered a risky service because it can spin up huge stacks of expensive hardware that too at a multi-region level for several accounts at once. Even though the service itself is free, users pay for the components spun up using CloudFormation. Any random template picked up online may have a complex structure and thus result in high cost usage. Because

of this, a template that is checked and under-control can have lots of users engaging and modifying it only to customize or enter private credentials and parameters.

6.1 Experiment1 - focusing on Elastic Beanstalk

The first experiment/trial was to use a template focusing on Elastic beanstalk. The steps were identical to the ones mentioned in the Configuration Manual. This trial did not turn out to be successful, the main result is always visible right after we create stack and then run it by customizing the various components. As seen in figure 2, when the stack was in creation, several internal errors resulted in complete rollback of the same, the final message also read **rollback completed**. This error is very common or the most common one in case a stack is not built.

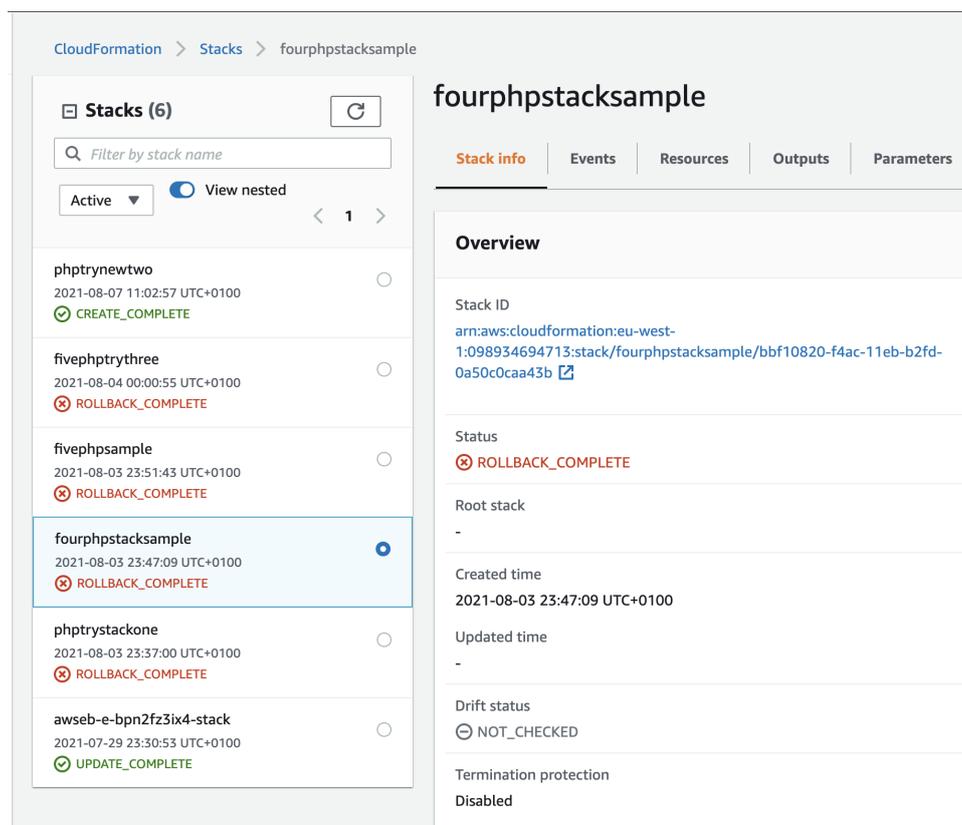


Figure 2: CFN fail

6.2 Experiment2 - Multiple services

The first experiment/trial and final trial had a few numbers in between, the final trial was the one which resulted in multi cloud-services from a single template. The steps were identical to the ones mentioned in the Configuration Manual. This trial turned out to be successful, many trials ran good for individual components but the real issue was to make different cloud services co-exist and be initiated from a single CloudFormation template. The approach was simple and pretty straight-forward, to start with one cloud service in a CloudFormation template and then keep incrementing a service count, continuous testing was necessary to recognize what service was not being setup the correct way.

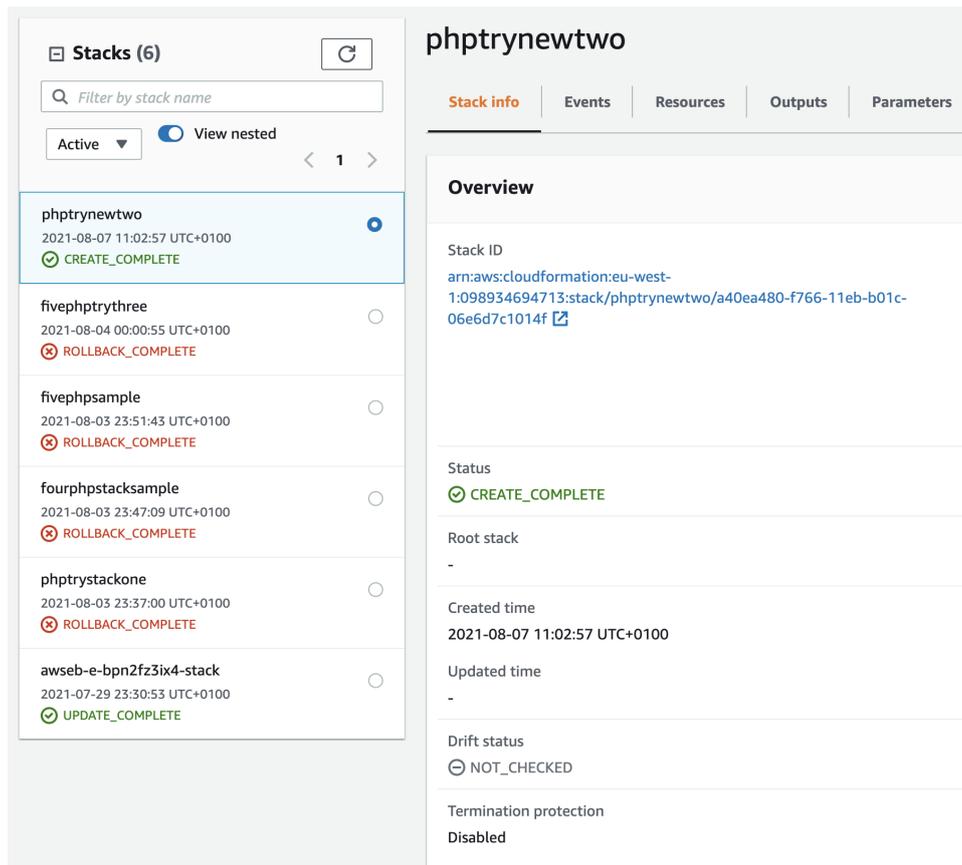


Figure 3: CFN Success

Eventually, a template was right enough to make some of the cloud services co exist. This particular combination is a treasure house for any application develop-ing/maintenance team. The stack created from this template not only had components such as EC2, S3, Elastic beanstalk, RDS co-exist, but also to carry Application load bal-ancers, auto-scaling group. This is what is a positive outcome and this has paved way for further research, a quest to add more and more services without hampering pre-existing ones and contributing towards making an application more reliable thus improving the user experience.

6.3 Discussion

- Experiment1 made it very clear from the start that the cloud services, their com-ponents need to be in-line with relevant parameters for things to go well.If not, then there will always be a failure in creation of the stack, although the stack was in process of creation, because of incorrect parameters, the entire process was rolled back by the AWS backend.
- Reaching the final test run wasn't a couple of steps from Experiment1. The main strategy was to make sure each and every single component would run separately using its personal parameters and script. Then, the components were added to a single template but one at a time. Regular testing was identical to unit testing and integrated testing. The testing was made at an incremental level, meaning that after each and every increment of a new service, the template was run to make sure

the integrated result until now runs smooth and as per expectations. The final test was judged as passed only when all services in the template were running and with desired features. An example of this is that the EC2 service was judged fine only when auto-scaling was found working.

- As pointed before, the template running is not the end goal, the main key is to make sure the application deployed on this template has changed the way the application would have reacted previously to any internal or external irregularities. The approach here is to use **random testing**, and we know the core component for an environment is the instance used to run and deploy the application. The best practice in case of instances is to ensure auto-scaling to meet any CPU usage and load-balancing to manage/divert any workload. To check, auto-scaling, we deliberately shut down an instance and see what happens next. Please note that the auto-scaling policy was mentioned in the template itself, where the minimum amount of instances running at any time was declared to 2.

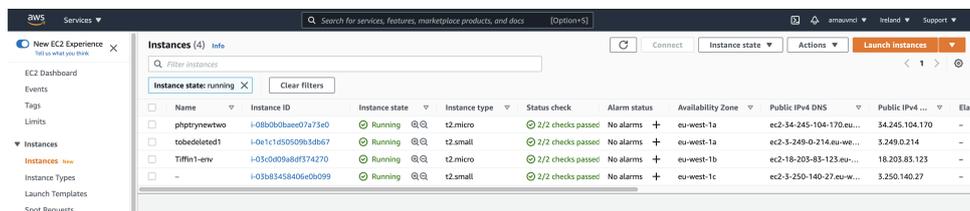


Figure 4: ASG1

In figure 4 we can see there are four EC2 instances running in the account. The instance which will be deleted to test auto-scaling has been renamed to **tobedeleted1**.

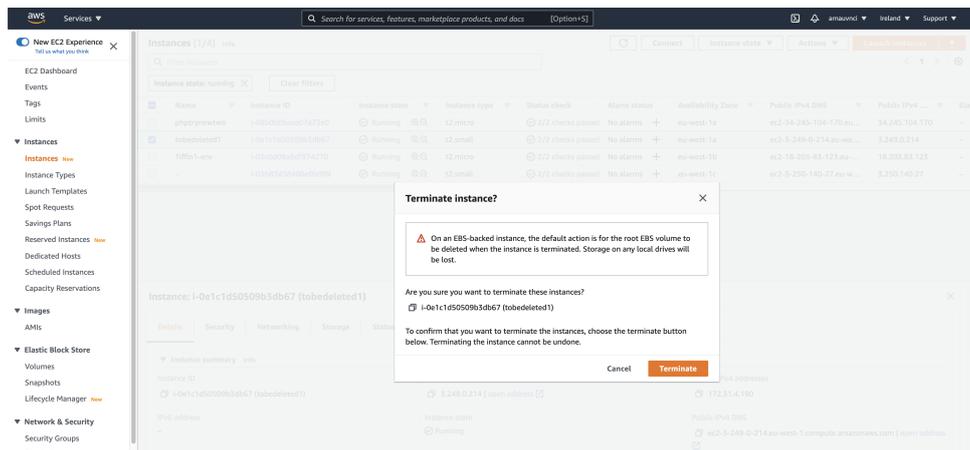


Figure 5: ASG2

In figure 5 we select the instance and terminate it.

In figure 6 we confirm that the target instance in shutting down, moreover, a new instance has been spun up automatically and is moving towards the usual running state.

In figure 7 although we see five instances, the target instance is in terminated state and hence we can ensure there are four instances. The relevant point here is that there are still two instances running that came out of running the CloudFormation stack.

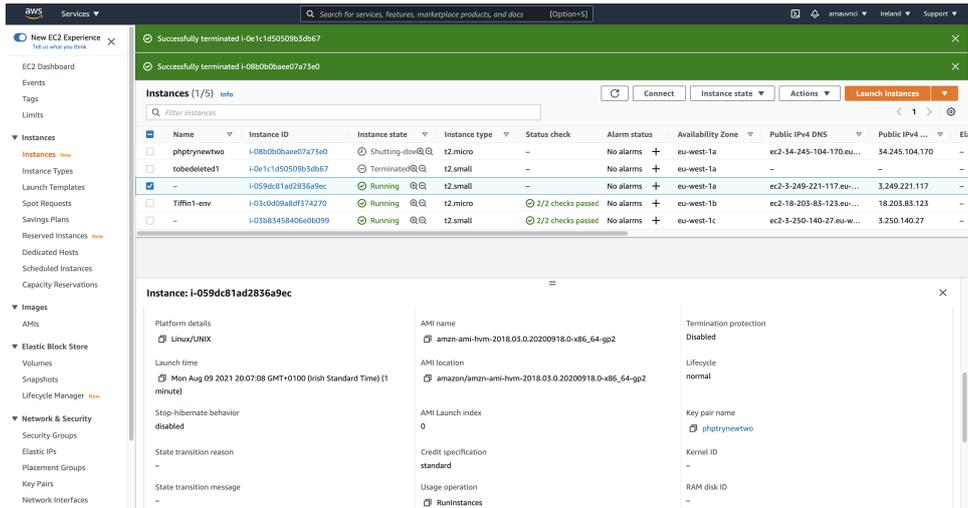


Figure 6: ASG3

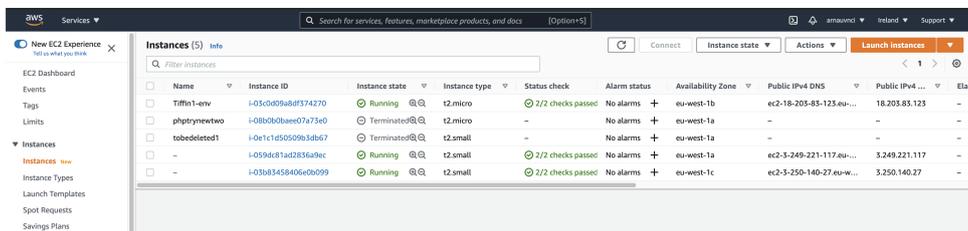


Figure 7: ASG4

6.3.1 Improvements

There is always a scope for improvement. In today's competitive world, average practices are rapidly becoming out-of-date. For this research, improvement is a clear need in case of failed experiment, but there is also room for improvement in case of the successful experiment discussed above. One can deep dive within existing cloud services to be able to configure them in more detail and hence have a better control on these services. A second one would be to add more services to existing stack. Finally, trying both will be the best option, adding more cloud services to the stack and that too having more detailed control over services will make most of the tiring provision over by running a simple template.

7 Conclusion and Future Work

This research proposes that there exists a need to use hardware and software in a synchronous way to avoid and be prepared in case of any calamity that may harm an application environment's smooth functioning.

The future work would be to add more and more cloud services possible while making sure they can co-exist. in a template successfully. Cloud services such as AWS Lambda and API Gateway are widely used today as serverless is picking up its pace in the market and so is consumption of internal and external APIs. A recommended future work would be to have a template using serverless services of Lambda and have API Gateway to consume APIs for an application. A point worth noting is that in case of API Gateway,

the parameters will be very critical and API calls and parameters and prone to human error and need accurate values/parameters to work.

A future template with more added services and having a granular access to associated components would definitely save a lot of time for infrastructure and development teams. Provisioning resources using a single template, being able to treat huge hardware stacks as a single entity, would save precious time in software development lifecycle, leaving room to focus on DevOps, best practices and even application maintenance.

References

- Alam, A. B., Haque, A. & Zulkernine, M. (2018), Crem: A cloud reliability evaluation model, *in* ‘2018 IEEE Global Communications Conference (GLOBECOM)’, IEEE, pp. 1–6.
- Babalola, A. A., Belkacemi, R. & Zarrabian, S. (2016), ‘Real-time cascading failures prevention for multiple contingencies in smart grids through a multi-agent system’, *IEEE Transactions on Smart Grid* **9**(1), 373–385.
- Benchara, F. Z. & Youssfi, M. (2021), ‘A new scalable distributed k-means algorithm based on cloud micro-services for high-performance computing’, *Parallel Computing* **101**, 102736.
- Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L. M., Netto, M. A. et al. (2018), ‘A manifesto for future generation cloud computing: Research directions for the next decade’, *ACM computing surveys (CSUR)* **51**(5), 1–38.
- Campbell, B. (n.d.), ‘The definitive guide to aws infrastructure automation’.
- Cui, T. & Li, S. (2020), ‘System movement space and system mapping theory for reliability of iot’, *Future generation computer systems* **107**, 70–81.
- Iac Tools list* (n.d.), <https://www.nexastack.com/en/blog/best-iac-tools>. Accessed: 2021.
- Kamal, M. A., Raza, H. W., Alam, M. M. & Su’ud, M. M. (2020), ‘Highlight the features of aws, gcp and microsoft azure that have an impact when choosing a cloud service provider’, *International Journal of Recent Technology and Engineering (IJRTE)* .
- Kantaria, M., Basilaia, G. & Chokhonelidze, G. (n.d.), ‘Development of the cloud services (aws) courses for the higher education institutions in georgia’.
- Lan, W., Zhou, K., Feng, J. & Chi, Z. (2010), Research on software cascading failures, *in* ‘2010 International Conference on Multimedia Information Networking and Security’, IEEE, pp. 310–314.
- Li, C., Song, M., Zhang, M. & Luo, Y. (2020), ‘Effective replica management for improving reliability and availability in edge-cloud computing environment’, *Journal of Parallel and Distributed Computing* **143**, 107–128.

- Li, X.-Y., Liu, Y., Lin, Y.-H., Xiao, L.-H., Zio, E. & Kang, R. (2021), ‘A generalized petri net-based modeling framework for service reliability evaluation and management of cloud data centers’, *Reliability Engineering & System Safety* **207**, 107381.
- Liu, X. X., Qiu, J. & Zhang, J. M. (2014), High availability benchmarking for cloud management infrastructure, *in* ‘2014 International Conference on Service Sciences’, IEEE, pp. 163–168.
- Liu, Z., Chen, Z., Liu, C., Sun, H. & Hu, Y. (2012), Multi agent system based wide area protection against cascading events, *in* ‘2012 10th International Power & Energy Conference (IPEC)’, IEEE, pp. 445–450.
- Mahmoud, Q., Andrusiak, I. & Altaei, M. (2018), Toward a reliable service-based approach to software application development, *in* ‘2018 IEEE 20th Conference on Business Informatics (CBI)’, Vol. 1, IEEE, pp. 168–177.
- Malatpure, A., Qadri, F. & Haskin, J. (2017), Experience report: Testing private cloud reliability using a public cloud validation saas, *in* ‘2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)’, IEEE, pp. 56–56.
- Mesbahi, M. R., Rahmani, A. M. & Hosseinzadeh, M. (2018), ‘Reliability and high availability in cloud computing environments: a reference roadmap’, *Human-centric Computing and Information Sciences* **8**(1), 1–31.
- Tian, Y., Tian, J. & Li, N. (2020), ‘Cloud reliability and efficiency improvement via failure risk based proactive actions’, *Journal of Systems and Software* **163**, 110524.
- van Vliet, J., Paganelli, F. & Geurtsen, J. (2013), *Resilience and Reliability on AWS: Engineering at Cloud Scale*, ” O’Reilly Media, Inc.”.
- Wang, H., Shen, H. & Li, Z. (2018), Approaches for resilience against cascading failures in cloud datacenters, *in* ‘2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)’, IEEE, pp. 706–717.
- Wittig, M. & Wittig, A. (2018), *Amazon web services in action*, Simon and Schuster.
- Xing, L. (2020), ‘Cascading failures in internet of things: review and perspectives on reliability and resilience’, *IEEE Internet of Things Journal* **8**(1), 44–64.