MSIS 2628
The Business of Cloud Computing

Submitted by Malya Jain
(W1279111)

Group #2 - GCP vs AWS

1. What are the PaaS or SaaS components that the two Cloud vendors provides. Pick one of the PaaS or SaaS offering and compare in terms of price, features, ease of use and benefits.

There are multiple core functionalities of the AWS SaaS and PaaS. AWS provides data warehouses and applications on its cloud service. It provides a large ecosystem to choose from and multiple applications based on the Hadoop Ecosystem.

PAAS Offering		
Functionality	**AWS**	**GCP**
Compute		
Core compute service	EC2 (Elastic Compute Cloud)	GCE (Google Compute Engine)
Machine Images	AMI (machine images	VMs spanned across availability groups and regions.
Compute Processes	Elastic Beanstalk	Google App Engine (Webapps)
Backend	Lambda	Firebase
Load Balancing	Across multiple availability zones and regions	Load Balancing available in addition to VM migration and multiple cores within the instances
Storage		
Storage type	Ephemeral storage for EC2	Temporary storage for GCE
Block Storage	EBS - Anonymous to hard disks in the sense that it can be attached to EC2 or kept in silos	Persistent disks equivalent to EBS
Permanent Storage	S3	Google Cloud Storage
Archiving Storage	Glacier	Nearline
Latency for recovery	Yes	No
BigData	DynamoDB, Kinesis	BigTable, Hadoop
Network		
Isolation	VPC (Virtual Private Network) – network topology, subnets, private IP address range	Subnet - Single network – gateway address range
Load Balancer	ELB (Elastic Load Balancer)	Load Balancing and Scaling
Pricing		
Types	On Demand, Reserved, Spot	Sustained use discounts
	Pay per second (minimum 1 min)	Pay per second (minimum 1 min)
	Reserved: Pay upfront. 1-3 instances	Committed use discounts
	Bidding system	No bidding system
Benefits		

Latency	Visible latency when restoring archived files	Zero latency when restoring archived files
Security	Data encrypted in transit	All data is encrypted
Future	Availability and redundancy	Rapid global expansion
Multitenancy	Very high multi-tenancy - no need to rewrite the software again	Moderate multi-tenancy depending on the data partitions also known as tenants
References	https://cloud.google.com/docs/compare/aws/compute https://blogs.endjin.com/2016/11/aws-vs-azure-vs-google-cloud-platform-networking/	

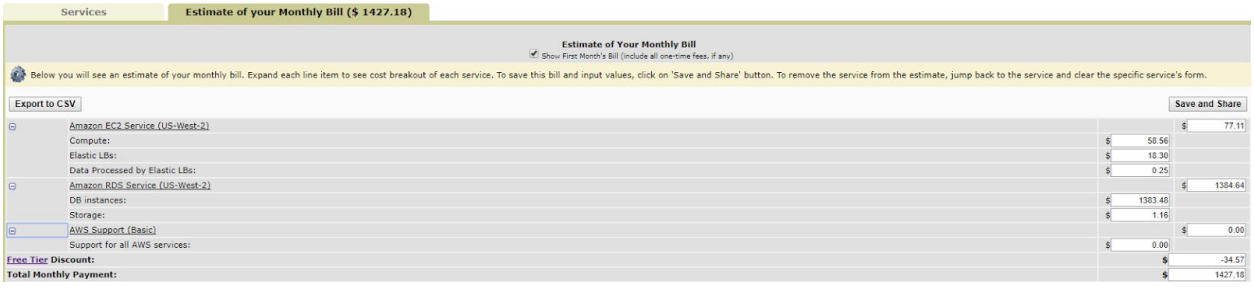
2. What would be the cost of the following configuration for the two Cloud vendors when run 24*7 for a whole year? How will you reduce the cost of such a deployment in the two cloud vendors you are working with ?

- 1 Load Balancer
- 4 EC2 instances
- 2 DB instances

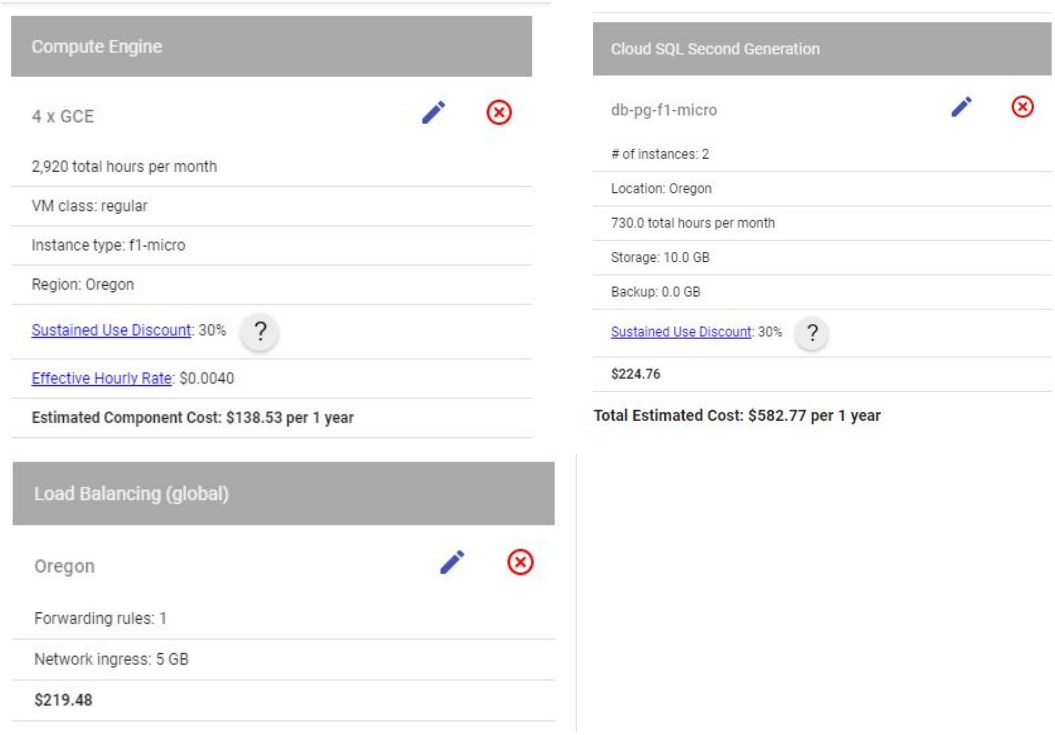
	AWS	GCP
1 Load Balancer	$0.028 per hour 0.028*24*365 = $245.28	$0.025 per hour 0.025*24*365 = $219
4 EC2 Instances	$0.0138 per hour 0.0138*24*365*4 = $483.55	$0.0053 per hour 0.0053*24*365*4 = $185.71
2 DB Instances	$0.022 per hour 0.022*24*365*2 = $385.44	$0.015 per hour 0.015*24*365*2 = $262.8
Total	$1,114.27	$667.51

Price calculated online:

AWS



GCP



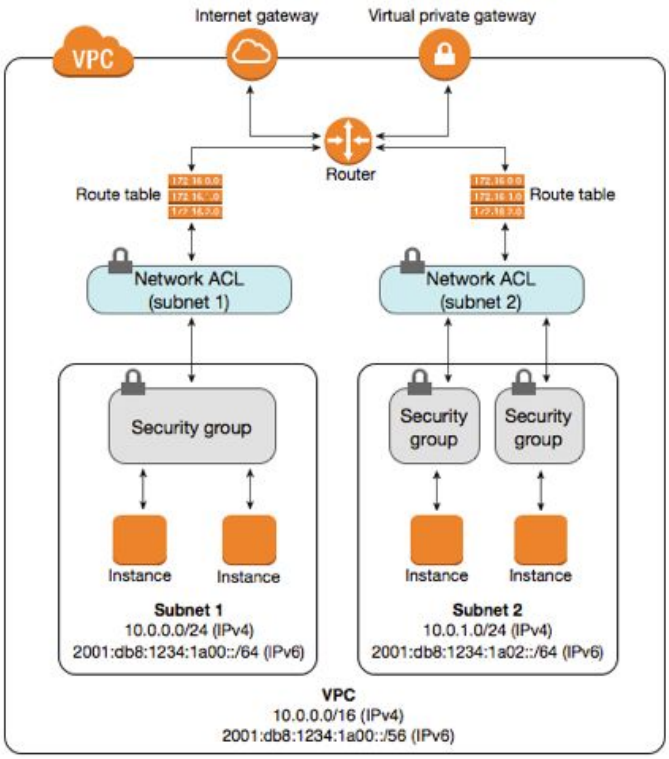
Reduce Pricing by taking the following steps:
AWS
Use Reserved Instances
Utilize Spot Pricing (bidding system can help reduce the cost) in case of AWS
Utilize the provision of Auto Scaling Groups so that the pricing is calculated only on the basis of usage. In case there is a spike in usage, more EC2 will spin up and in case the load is low, the EC2 will be terminated
GCP
Utilize multiple scheduling softwares
Use GCP's pre-emptible virtual machines (VMs)
Switch off the development and testing instances when not in use
Use Compute Engine AutoScaler so that the demand is fulfilled when required

3. Compare the Security offering of the two Cloud vendors. Compare VPC, ACL and Security Groups that is provided by the two vendors. Which one is more secure and why? Give references, screen shots to support your answer.

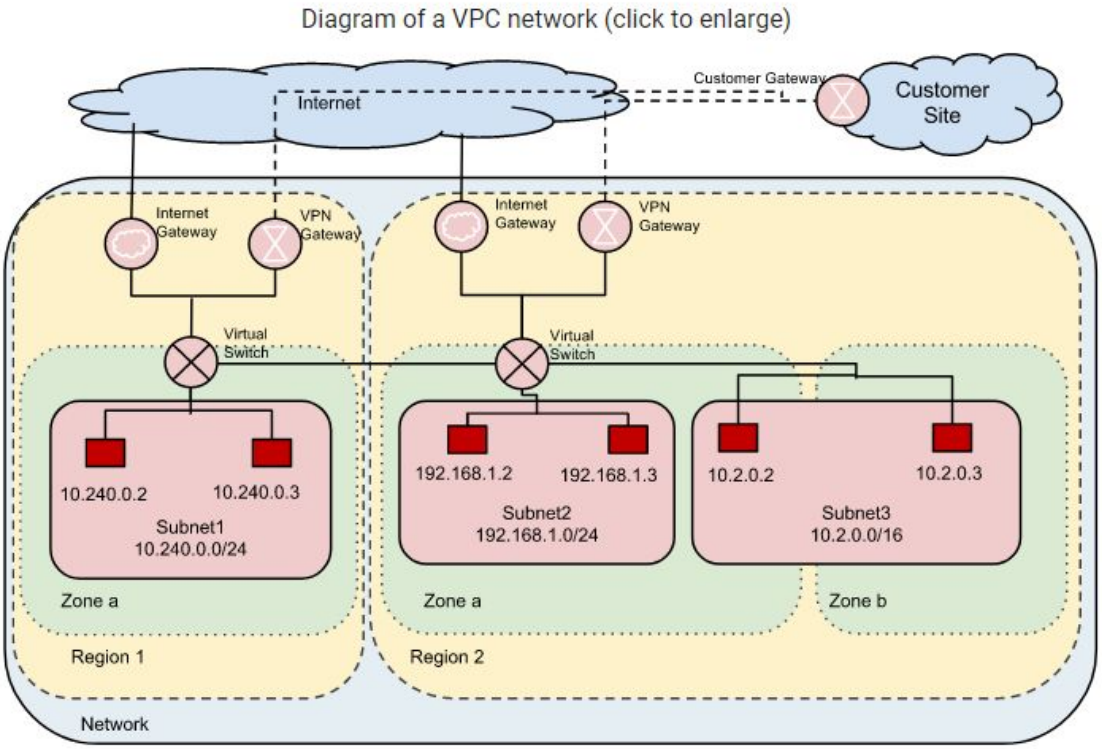
Functionality	AWS	GCP
Data at rest and in motion	Data encrypted at rest and transit	Data encrypted at rest and transit
Key Management	PEM key required to spin up instance	KMS (Key Management System) uses AES256 keys
Detection and Prevention	IDS (Intrusion Detection System) IPS (Intrusion Prevention System)	Intrusion Detection by laser beam IPS at data entry points
Subnets	Restrict IP ranges in private IP subnets	Do not restrict private IP ranges for subnets
Firewall	WAF (Web Application Firewall) Security Groups	Ingress and Egress firewall rule detection Compute Engine Firewall Rules
IAM (Identity and Access Management)	Segregation of Duties based on roles	Granular level with Audit Logs enabled
Multifactor Authentication	Multi-factor Authentication	Security Key Enforcements Only available with G Suite tools
Encryption	Stored data is not by default encrypted	All gcloud data in storage is by default encrypted
Encryption Methodology	BYOK (Bring Your Own Key)	API Encryption and Decryption
VPC (Virtual Private Cloud)	Select own IP range	• Virtual Network Partition • Kubernetes engine used for VMs • Types: Auto Mode and Custom Mode
Firewall	Default - Incoming: All blocked Outgoing: All allowed	Default - Incoming: All blocked Outgoing: All allowed
ACL (Access Control List)	• Operation is at Subnet level which provides second layer of defense • Allow and Deny rules	• Can restrict access to internal objects and buckets • By default - Project Viewers, Editors, owners

	• Stateless	
SG (Security Groups)	• Operation is at instance level which provides first layer of defense • Allow rules only • Stateful	• Does not offer Security Groups as such • Ingress and Egress Firewall settings
Who is more secure? Give Reasons	Both AWS and GCP maintain exceptionally high standards of data security and are compliant to all the data security standards such as PCI, HIPAA, SSAE 16. Both the companies are leaders in providing security solutions. Hence, there is no clear winner in the security aspects. Both are equally ahead in their security. The comparison may differ in case of very specific business requirements (for example federal data) that may yield a clear choice of the two.	
References	http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Security.html	https://cloud.google.com/vpc/docs/vpc https://cloud.google.com/docs/compare/aws/compute

AWS VPC diagram



GCP VPC Diagram

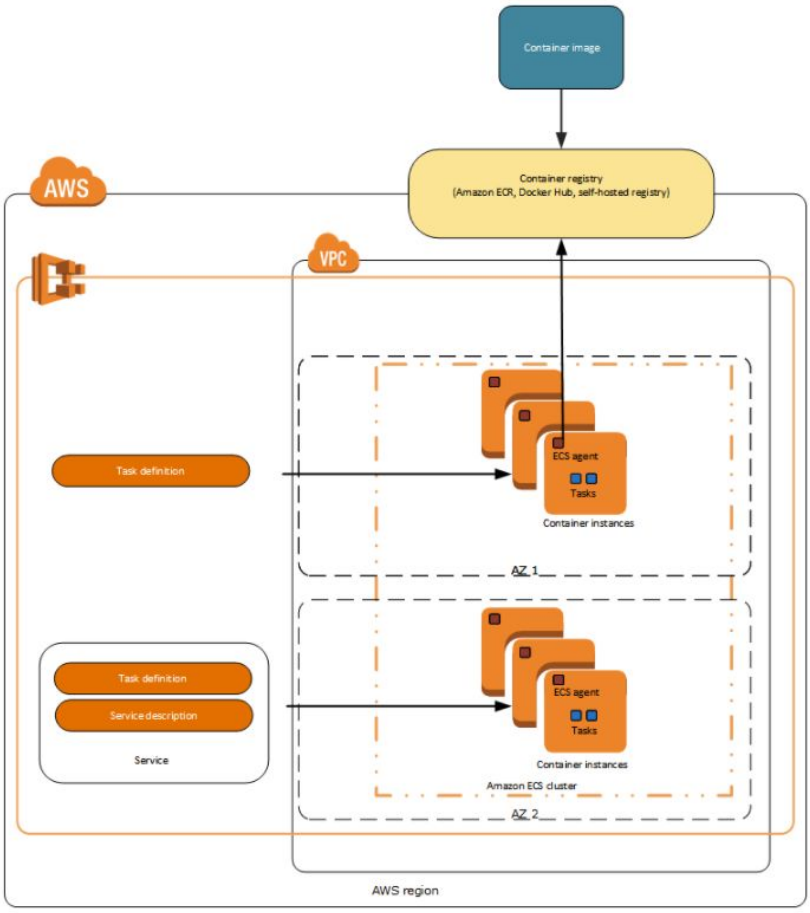


4. How is Docker supported in the two Cloud Vendors ? Is one vendor better or more friendly for Docker repositories ?

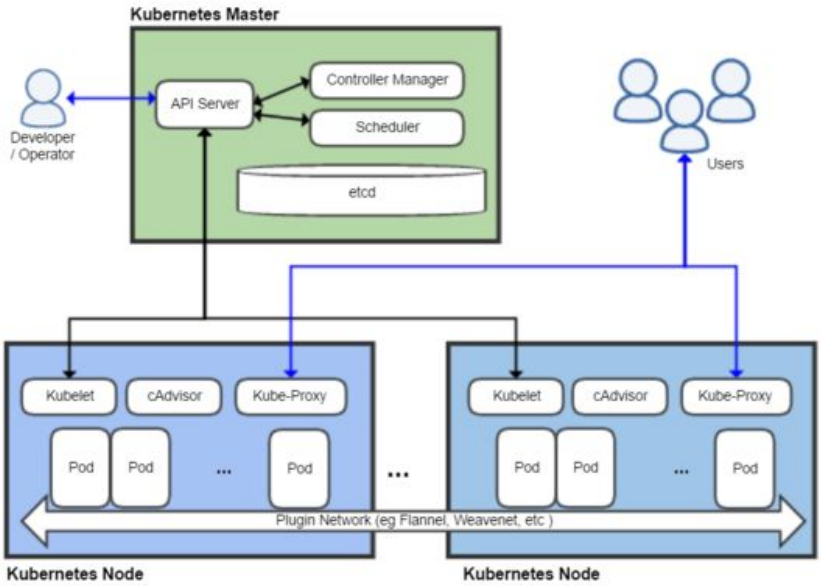
Functionality	AWS	GCP
Dockerization	• AWS Docker • Part of EC2 Container Service • Standardized unit used for Software Development. Provides everything that one needs for code starting from runtime, system tools to managing system libraries, etc. • Created from an Image which is read-only template • Very high scalability • Simple calls to API can start or stop the docker service • Easy integration with other AWS Services • Highly secure and high reliability • Data Processing and Container	• Google Kubernetes Engine (GKE) • Software dependencies are removed • Application deployment is rapid • For stateless applications as well as databases by attaching persistent storage • Automatic provision of underlying cloud resources • Effortless scaling provided • Application support - stateless • 0-5 nodes are free • Nodes utilize the container optimized image • Functionality of auto upgrade and auto repair • Fully managed by SREs (Site Reliability Engineers)

	are provided as a service	
Which one is better?	This argument is open ended as the qualities depend purely on the business use case and the application that needs to be installed, especially depending upon the SDKs. However, if I were to give my opinion, I'd say that Google Dockerization is more friendly than AWS Docker. I am providing this judgement basis my own research and also taking inputs from the guest lecture on Google Cloud Platform by Sowmya Kannan. Reasons are as follows: ● Google Kubernetes is comparatively more cohesive ● It has a more cluster centric view ● As mentioned above, the SREs constantly monitor the dockers for any activity that requires attention	
Infra	Can be deployed on EC2 only	Can be deployed on any physical or virtual hardware, even on public clouds Even compatible with AWS, RedHat and Azure
Availability	Depends on the service requirements	HA - High Availability Pods are monitored constantly and removed if detected to be unhealthy
Scalability	No specific figures available for the scalability of the nodes	Can be scaled upto 1000 nodes
Response	The response has some latency. Not as fast as GKE	99% of API calls are returned in less than one second
Conclusion	GKE is more than just a docker container service, hence better than AWS Docker	

AWS Docker



GKE



5. What tools does the Cloud Vendors provide to help with Automation like AWS CLI? If not similar to AWS CLI do they provide any APIs to help with Automation? Compare and contrast the automatin tools provided by the two Cloud vendors.

GCP features and tools

• Cloud SDK - Software Development Kit
• gloud used to manage VMs (create, start and manage)
• Client libraries available for python, php, ruby
• Can simulate the development, test and validation environment
• gsutil tool allows to manage cloud storage buckets and associated objects
• bq tool is used to manipulate datasets and associated tables
• The API Client Library provided by GCP are as follows:
  ○ Java and JavaScript
  ○ Python
  ○ PHP (Beta)
  ○ .NET
  ○ Objective-C
  ○ Dart (Beta)
  ○ Ruby (Alpha)
  ○ Go (Alpha)
  ○ Nodejs (Alpha)

Comparisons

Functionality	**AWS ECS CLI**	**GCP SDK (kubectl or gcloud)**
Availability	Need to configure the AWS CLI. It is not available automatically.	No need to configure to start using it. It is automatically available
Management	Nodes health is evaluated when the health check is done	Creating clusters and its functioning is related to the nodes' health
Function	The API distinction is not very clear. It comes across as unavoidably bubbling.	Cluster orchestration system which is quite clear and obvious in the API tools that it offers
Commands	First the login is required by using this command *aws ecr get-login* then *docker login* and then *aws docker push*	Very simplistic such as for pushing a new image to the docker, it is *gcloud docker push*
Reference: https://medium.com/@betz.mark/comparing-amazon-elastic-container-service-and-google-kubernetes-1c63fbf19ccd		

6. Research and compare the the Uptime (in 4 or 5 nines) and SLA offered by the two Cloud vendors. How do the Cloud vendors report down times? If the Cloud Vendors have data centers around the globe, compare the global data center presence and the Uptime for the various data centers around the globe.

Functionality	AWS	GCP
Monthly Uptime	99.99% (EC2 and EBS) 99.95% (RDS) 99-99.9% (S3 - depends on the service) 100% Route53	99.95% (GCE) 99.99% (Cloud Storage)
Potential downtime per year	EC2 - 04:23 S3 - 15:39 to 08:46	GCE - 04:23 Cloud Storage - 08:46
Service Credit	10% if the uptime > = 99% but < 99.99% 30% if the uptime < 99%	10% if the uptime ~ (99 - 99.95) % 25% if the uptime ~ (95.0 - 99) % 50% if the uptime < 95%
Exception	Does not take into consideration downtime due to third party dependencies like software or hardware equipment	For factors which lie outside of Google's control or the customer behavior seem to violate the contract agreement
Maintenance periods	5 mins	< 5 mins
Failover	Partial automatic failover to other servers	Live migration and replication which does not affect the cloud server availability
Regions	16	13
Availability Zones	44	39
References	https://www.cloudorado.com/vs/aws_vs_google	

Reliability & Failover

	amazon web services	Google
SLA level	99.995%	99.95%
SLA credit	10% of annual bill	10% - 50%
SLA exceptions	5 min.	maitenance, periods < 5 min.
Multiple zones in region	✓ ⓘ	✓
Tier 3+ DC		
Automatic failover to other server	partial ⓘ	✓ ⓘ
Storage attachment	Instance storage: server EBS: network	network & local
RAID level		
Backup - snapshot	✓	✓ ⓘ
Backup - storage	✓ ⓘ	✓

Global Data Centers AWS

AWS locations. Diagram by Amazon

Global Data Centers GCP

Google Cloud locations

Description of the downtimes depending on the number of nines

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	**36.5 days**	**72 hours**	**16.8 hours**	**2.4 hours**
95% ("one and a half nines")	18.25 days	36 hours	8.4 hours	1.2 hours
97%	10.96 days	21.6 hours	5.04 hours	43.2 minutes
98%	7.30 days	14.4 hours	3.36 hours	28.8 minutes
99% ("two nines")	**3.65 days**	**7.20 hours**	**1.68 hours**	**14.4 minutes**
99.5% ("two and a half nines")	1.83 days	3.60 hours	50.4 minutes	7.2 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes	2.88 minutes
99.9% ("three nines")	**8.76 hours**	**43.8 minutes**	**10.1 minutes**	**1.44 minutes**
99.95% ("three and a half nines")	4.38 hours	21.56 minutes	5.04 minutes	43.2 seconds
99.99% ("four nines")	**52.56 minutes**	**4.38 minutes**	**1.01 minutes**	**8.64 seconds**
99.995% ("four and a half nines")	26.28 minutes	2.16 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	**5.26 minutes**	**25.9 seconds**	**6.05 seconds**	**864.3 milliseconds**
99.9999% ("six nines")	**31.5 seconds**	**2.59 seconds**	**604.8 milliseconds**	**86.4 milliseconds**
99.99999% ("seven nines")	**3.15 seconds**	**262.97 milliseconds**	**60.48 milliseconds**	**8.64 milliseconds**
99.999999% ("eight nines")	**315.569 milliseconds**	**26.297 milliseconds**	**6.048 milliseconds**	**0.864 milliseconds**
99.9999999% ("nine nines")	**31.5569 milliseconds**	**2.6297 milliseconds**	**0.6048 milliseconds**	**0.0864 milliseconds**

AWS SERVICES	MONTHLY UPTIME PERCENTAGE	SERVICE CREDIT	MONTHLY UPTIME CALCULATION	COMMENTS
EC2 Instances & EBS Volumes	< 99.95%	10%	100% – EC2 or Amazon EBS was in state of "Region Unavailable"	"Region Unavailable" and "Region Unavailability" mean that more than one Availability Zone in which you are running an instance, within the same Region, is "Unavailable" to you. "Unavailable" and "Unavailability" mean: For Amazon EC2, when all of your running instances have no external connectivity. For Amazon EBS, when all of your attached volumes perform zero read-write IO, with pending IO in the queue.
	< 99%	30%		
S3 (storage services)	< 99.9%	10%	100% – Error Rates	"Error Rate" means: Total No. of internal server errors / Total number of requests for applicable request during that five minute period
	< 99%	30%		
S3 (storage services) Standard – Infrequent Access (Standard-IA)	< 99%	10%	100% – Error Rates	
	< 98%	30%		
CloudFront (CDN service)	< 99.9%	10%	100% – Error Rates	
	< 99%	25%		
RDS (DB service)	< 99.95%	10%	100% – percentage of 1 minutes period multi-AZ was unavailable	"Multi-AZ instance" means an Amazon RDS for MySQL, MariaDB, Oracle or PostgreSQL database instance with the Multi-AZ parameter set to true. "Unavailable" means that all connection requests to the running Multi-AZ instance fail during a 1 minute period.
	< 99%	25%		
Route 53 (DNS service)	5 – 30 minutes	1 day	Duration was not 100% available	
	31 mins – 4 hours	7 days		
	More than 4 hours	30 days		

7. Compare and contrast Auto Scaling Group for the two Cloud vendors, in terms of cost and features?

Functionality	AWS	GCP
Terminology	Auto Scaling	Autoscaler
Features	• Monitor health of instances which are running • Repair the instances which are impaired • Capacity is automatically balanced between zones	• For managed instance groups • Can be created from a common instance template • Autoscaling policy should be defined on the basis of avg CPU utilization, sub queueing workload etc
Auto Scaling	Associate ELB with auto scaling, which is another service in the platform. Need to be scaled manually.	GCE balancers automatically scale depending upon the spike
Pricing	Free of cost. Bill is based on CloudWatch usage	Free of cost. Based on usage ~ 30-35% cheaper than AWS
Dynamic Scaling	For AWS Fleet	For cluster autoscaling - kubernetes should be used
Monitoring	• CloudWatch • Manually notifications can be set up	
Storage	Not so effective in distribution of data to large workforce	Persistent disk are allowed to be attached to instances (read only mode)

8. How does the AWS S3 compare or Object Storecompare to the features that the two Cloud Vendors provides related to Cloud Storage? Which one would you prefer and why? How do the costs compare?

Functionality	AWS	GCP
Upload Mechanism	Via Java based API	Via Google's Command Line Utility
Preference	Google Cloud Storage because of the following reasons:	
Upload big files	On Google cloud it is 4x faster	

Uploading small chunks files	On Google Cloud it is 20x faster
Cost	On Google Cloud it is 35% cheaper

Object Store	Storage cost (cents/GB/mo)	Egress cost (cents/GB)	Ingress cost (cents/GB)	Availability %
Amazon Glacier (US standard)	0.4	9.0	0.0	99.99
Amazon S3 Infrequent Access (US standard)	1.25	10.0	0.0	99.9
Amazon S3 Reduced Redundancy (US standard)	2.4	9.0	0.0	99.99
Amazon S3 Standard (US standard)	2.3	9.0	0.0	99.99
Google Cloud Coldline Storage	0.7	12.0	0.0	99.0
Google Cloud Multi-Regional Storage	2.6	12.0	0.0	99.9
Google Cloud Nearline Storage	1.0	12.0	0.0	99.0
Google Cloud Regional Storage	2.0	12.0	0.0	99.0

9. How does the Compute Engine like EC2 cost, features compare for the two Cloud Vendors? Which one would you prefer and why?

Features	**AWS**	**GCP**
Name	EC2 (Elastic Compute Cloud)	CE (Compute Engine)
Instances	39	18
Acceleration	Yes (GPU)	No (GPU)
Can create a custom instance?	No	Yes
Memory statistics	0.5 - 244 (GB)	0.6 - 208 (GB)
Storage	48 Terabytes	3 Terabytes
Availability	Easy and automatic scaling	Can do horizontal scaling
OS	All major OS are available (11)	9 OS available except CloudLinux and Oracle Linux
DB	All major DB available (5+)	3 DBs available, except MS SQL Server, Oracle and Maria DB
Pricing	Free to use up to a certain limit (750 hours). Pricing depends upon	Free to use up to a certain limit (744 hours). Pricing depends on machine

	On demand, spot and reserved. Per second pricing is also available	type pricing. Per second billing is also available
Preference	I would choose AWS EC2 over GCP CE as a preference for me personally because of the following reasons: ● AWS is a market leader with the longest standing amongst the cloud providers ● AWS EC2 functionalities are more mature ● Official AWS world class and simple to follow documentation along with tutorials are available for each and every feature on AWS EC2 ● All major OS and DB are available within AWS ● Autoscaling is advanced in EC2 as compared to GCP ● Provides higher SLA and higher refund in case the downtime exceeds the SLA	
References	https://aws.amazon.com/ec2/pricing/ https://cloud.google.com/compute/pricing	

10. **If you had to describe one really Cool feature of the Cloud Vendors what would that be ?**

AWS	After reading and working on the AWS platform, the coolest feature of AWS as per me is a relatively new feature of IOT integration. It is just a very simple button which can be used by the users to create their applications and install them easily. The functionality is called Amazon Dash buttons. It provides fully functional features of AWS on the top of that, it is hosted on the AWS console. It is a pre programmed application which is provided as a front end to the users.
GCP	GCP provides its coolest feature of Kubernetes, which I feel is the best feature provided by GCP. The cloud providers and users rely heavily on the container services because of its ease of use and functionalities.

Scenario and Scope

The scope consists of **hypothetical medical application** called - MEDOPS used by the ABC Hospitals in USA. The Hospitals are located at over 50 locations across the country. The The application is capable of the following functionalities:

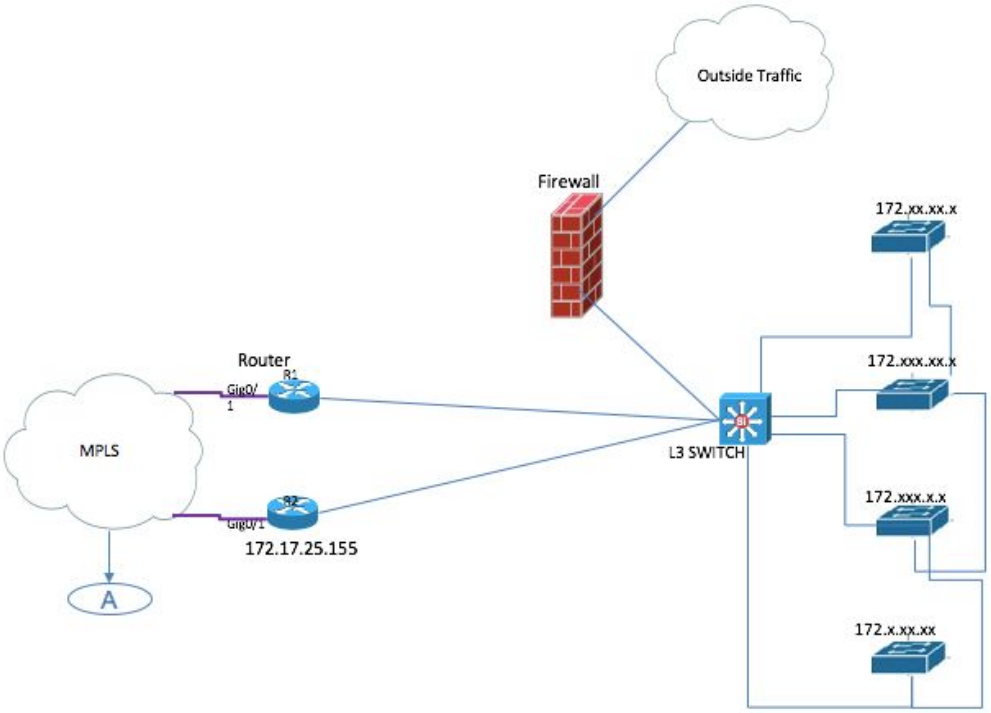
- The in-patient record is entered when the patient arrives at the hospital.
- A unique Patient ID (PID) is created upon first entry of the user. The PID depends on the unique values of Name, Date of Birth and Postal Code of the user.
- Upon every visit of the patient, a unique record is created in the database however, linked to the same PID
- There are modules in the application which are linked to multiple departments depending on the services provided to the patient
- When the doctor diagnoses a patient, he/she can pull out the history of the patient only by the Patient ID

Current landscape of the MEDOPS application

- The application has been developed inhouse and deployed locally in the DataCenter
- Each hospital location has an IT Team consisting of an Incident Responder, Network Analyst and Data Center Analyst

- Each hospital location also consists of a small server room or datacenter where the application is hosted.
- All the applications data is synced to the private cloud of the hospital with a batch job which runs at midnight everyday

Current Network Diagram per server room



Local IT Infrastructure consists of the following

1. Application Development team, which develops and deploys the application at multiple locations
2. Security Team which monitors the security of the IT Infrastructure. This team also makes sure that the application is HIPAA compliant and manages the internal and external audit procedures
3. Procedures are defined and implemented to maintain an inventory of stored and archived media to ensure their usability and integrity. Asset Inventory per region and managed

locally at the HQ which contains the details such as Asset Type, Asset Tag, Asset ID, Serial Number, OS, Location, Model Number etc.

4. Expensive and over the shelf data encryption mechanisms have to be purchased and deployed in order to comply with the HIPAA regulations
5. Datacenter and Server Rooms (per location)
    a. Air Conditioning (AC) Units, temperature and humidity controls, are present within the Data Centre, and are monitored and tested on a periodic basis.
    b. Periodic maintenance to ensure that cabling are well secured, structured and organized. False flooring and false ceiling are built so that wiring is not left out open
    c. Smoke detectors, heat detectors, fire extinguishers, and a gas based fire suppression system are in operation within the Data Centre
    d. CCTV Cameras
    e. Physical access procedures are defined to provide, limit or revoke physical access to the facility areas and restricted according to business needs
    f. 24x7 monitoring of the datacenter
6. Backup and Recovery team which takes the backup every day. An offsite location is be defined for offsite movement of backup tapes.

These are the challenges that the company is facing for hosting the application on-premise on private cloud

1. As the application is synced to the HQ Datacenter once a day only, the patient record is not synced real time. This brings us to a use case that if a Doctor, located in California, wants to discuss the urgent diagnosis of the patient with a Doctor in Texas, the data pulled from the application will not be consistent as the data is not synced until midnight.
2. Inconsistent and ineffective assessment of existing IT systems is leading to inability to meet business needs and expectations.

3. HIPAA regulations consist of strict guidelines. The requirements are identified, however, the assurance completely relies on the IT Function. The organization hires a third party to be audit-ready which is expensive
4. The application source documentation has to be secured with appropriate segregation of duties
5. Security architecture is not appropriately defined to implement security controls with respect to the access from different channels such as mobility and internal LAN
6. Super-user accounts are not restricted to authorized administrative personnel, it is available with everyone in the IT Team with no audit logging
7. Password parameters are not adequately defined
8. Controls relating to access to program and data are faulty with inappropriate documentation
9. Redundancy shall be defined and implemented for critical IT and network components
10. IT Continuity plan/BCP & DR Plan are not defined and documented.

If the company decides to utilize the AWS Cloud Infrastructure for its IT Services, following are the key benefits: (PROS)

1. By switching over the infrastructure to AWS, the company is assured agility and extensive collaboration.
2. AWS is extremely advanced in recognizing the security and privacy requirements and compliance to the regulations of HIPAA, FedRAMP, NIST 500-83, ISO 27001
3. By deploying the application on AWS, the time and effort will reduce significantly to run the analytics dashboard and workloads
4. The dependencies on managing the datacenter will reduce significantly by deploying on AWS instances because it takes the model of pay as you go and no additional expenditure on the devices.
5. AWS currently has more than 1800 controls restricted to just security, hence the question of security is well answered there.

6. There are so many other offerings provided by AWS that can be leveraged for the application deployment
7. It is also beneficial when the IT growth does not happen as per the plan and the underlying infrastructure goes for waste
8. There is no dependency at all for managing data center and all the operating expenditure spent on managing the data center can be saved and spent on AWS Marketplace
9. With AWS, the security and privacy policies and procedures are defined as per the global best practices with no place for slacking
10. It takes care of the administrative, technical and physical controls
11. With AWS, the access to program and data is very tightly controlled and audit logs are available for monitoring of the user activity. Any unusual or suspicious activity can be identified and triaged effectively on time
12. PHI - Personal Health Information is the most critical data that is required to have strict security and privacy controls can be managed more effectively on AWS rather than in-house infrastructure and policies
13. AWS does not technically hold the HIPAA certification because the cloud providers are not eligible for it, however, AWS meets the requirements underlying in HIPAA by aligning its practices to HIPAA risk control matrix
14. The AWS datacenters have very strong physical and logical access controls which assure more security than the in-house data center setup
15. The PHI is encrypted at rest and in transit. AWS has a HIPAA specific addendum called Business Associate Addendum (BAA)
16. The hospital may sign this addendum and then can be eligible for encryption for services eligible for HIPAA
17. The key management feature is extremely helpful in AWS where it is easy to manage and audit
18. The hospital should adopt EC2 which is scalable and can be configured easily with minimum support.

19. Once on AWS, the hospital need not worry about managing patches at the OS level, managing the Configuration Management Database
20. Amazon Elastic Beanstalk provides a unique encryption key which can be provided to the customer as an asymmetric key
21. Amazon S3 can be used as a data repository and encrypt the data at rest so that the management of the data is also taken care of. The S3 is free to use and minimal payment upon additional functionalities
22. The MEDOPS database can be taken as the Amazon RDS because again it provides the functionality of ecrypt the specific fields in the table
23. Load balancers can be set up (classic load balancer in this case) so that the load can spin up and shut off the instances at the time of loads
24. Easy access will be provided to the users even on iPad at home when a Doctor wants to refer to the patient record
25. Dashboards and reports will be created fast
26. SLA is fixed and guaranteed in case of AWS, hence, there will be almost zero downtime
27. Maintenance and operating costs are reduced significantly

Mechanics

- How to replicate the existing system to cloud service
- SWOT Analysis of the cloud vendors available in the maret and then finalizing which provider suits the requirements in the best possible way
- An intensive cost analysis will have to be performed with all the features taken into consideration and see how the capital is saved
- How many EC2 instances to spin up?
- What kind of load balancer to deploy to better suit the requirements?
- New employees with AWS skill will have to be recruited

Migration methodology

1. Conduct an in depth financial assessment

2. Assessment of the Security compliance and privacy controls. The most important question that should be evaluated is “Are strict controls present for CIA (Confidentiality, Integrity and Availability)
3. Create an architecture after conducting a technical assessment
4. Identify what are the dependencies of the application such as upstream and downstream dependencies.
5. Do a proof of concept assessment
6. Measure success criteria
7. Create migration strategy and phases of testing and deployment
8. Migrate the application

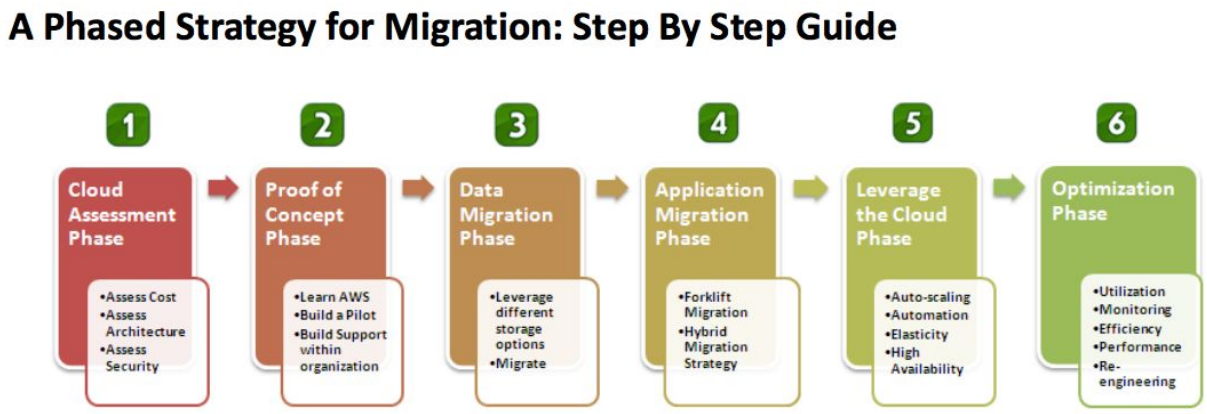


Figure 1: The Phase Driven Approach to Cloud Migration

Phases	Benefits
Cloud Assessment • Financial Assessment (TCO calculation) • Security and Compliance Assessment • Technical Assessment (Classify application types) • Identify the tools that can be reused and the tools that need to be built • Migrate licensed products • Create a plan and measure success	**Business case for migration (Lower TCO, faster time to market, higher flexibility & agility, scalability + elasticity)** **Identify gaps between your current traditional legacy architecture and next -generation cloud architecture**
Proof of Concept • Get your feet wet with AWS • Build a pilot and validate the technology • Test existing software in the cloud	**Build confidence with various AWS services** **Mitigate risk by validating critical pieces of your proposed architecture**
Moving your Data • Understand different storage options in the AWS cloud • Migrate fileservers to Amazon S3 • Migrate commercial RDBMS to EC2 + EBS • Migrate MySQL to Amazon RDS	**Redundancy, Durable Storage, Elastic Scalable Storage** **Automated Management Backup**
Moving your Apps • Forklift migration strategy • Hybrid migration strategy • Build "cloud-aware" layers of code as needed • Create AMIs for each component	**Future-proof scaled-out service-oriented elastic architecture**
Leveraging the Cloud • Leverage other AWS services • Automate elasticity and SDLC • Harden security • Create dashboard to manage AWS resources • Leverage multiple availability zones	**Reduction in CapEx in IT** **Flexibility and agility** **Automation and improved productivity** **Higher Availability (HA)**
Optimization • Optimize usage based on demand • Improve efficiency • Implement advanced monitoring and telemetry • Re-engineer your application • Decompose your relational databases	**Increased utilization and transformational impact in OpEx** **Better visibility through advanced monitoring and telemetry**

Success Criteria	Old	New	Examples on How to Measure
Cost (CapEx)	$1M	$300K	60% savings in CapEx over next 2 years
Cost (OpEx)	$20K/Year	$10K/Year	Server-to-Staff ratio improved by 2x 4 maintenance contracts discontinued
Hardware procurement efficiency	10 machines in 7 months	100 machines in 5 minutes	3000% faster to get resources
Time to market	9 months	1 month	80% faster in launching new products
Reliability	Unknown	Redundant	40% reduction in hardware-related support calls
Availability	Unknown	At least 99.99% uptime	20% reduction in operational support calls
Flexibility	Fixed Stack	Any Stack	Not locked into particular hardware vendor or platform or technology
New opportunities	10 projects backlog	0 backlog, 5 new projects identified	25 *new* projects initiated in 3 months

Table 2: Examples on how to measure success criteria

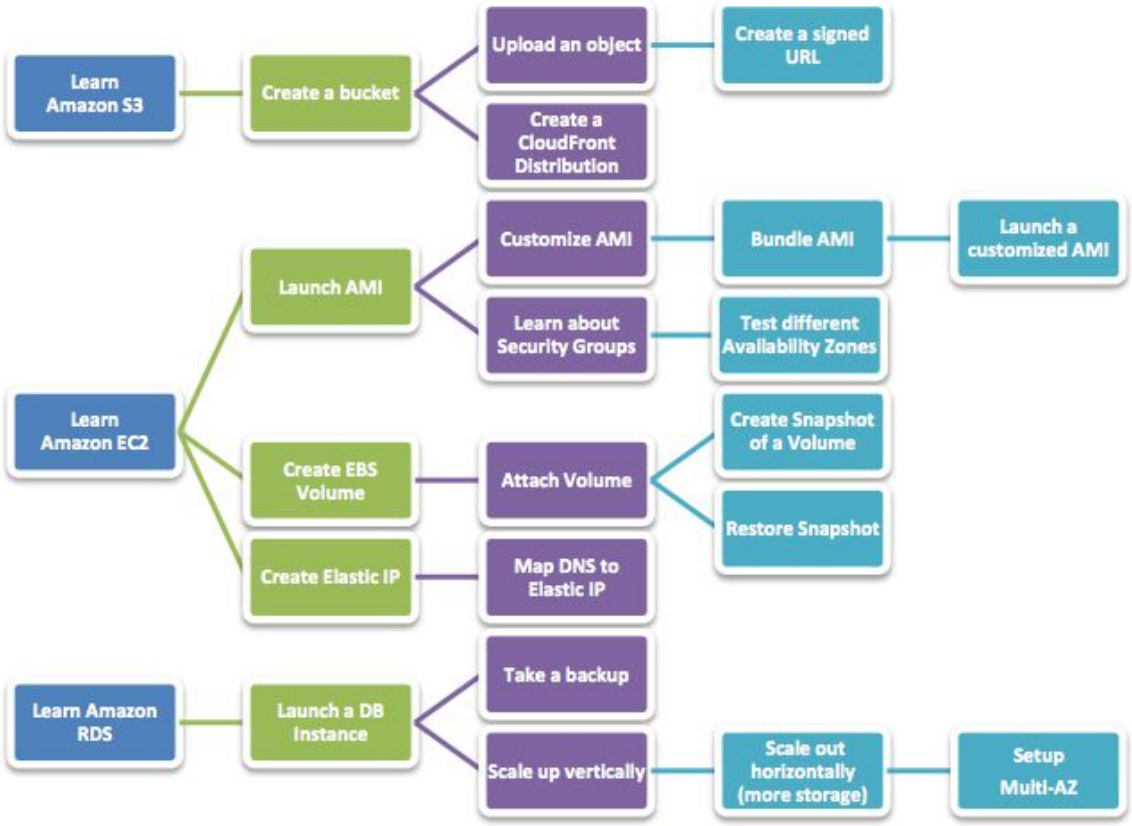
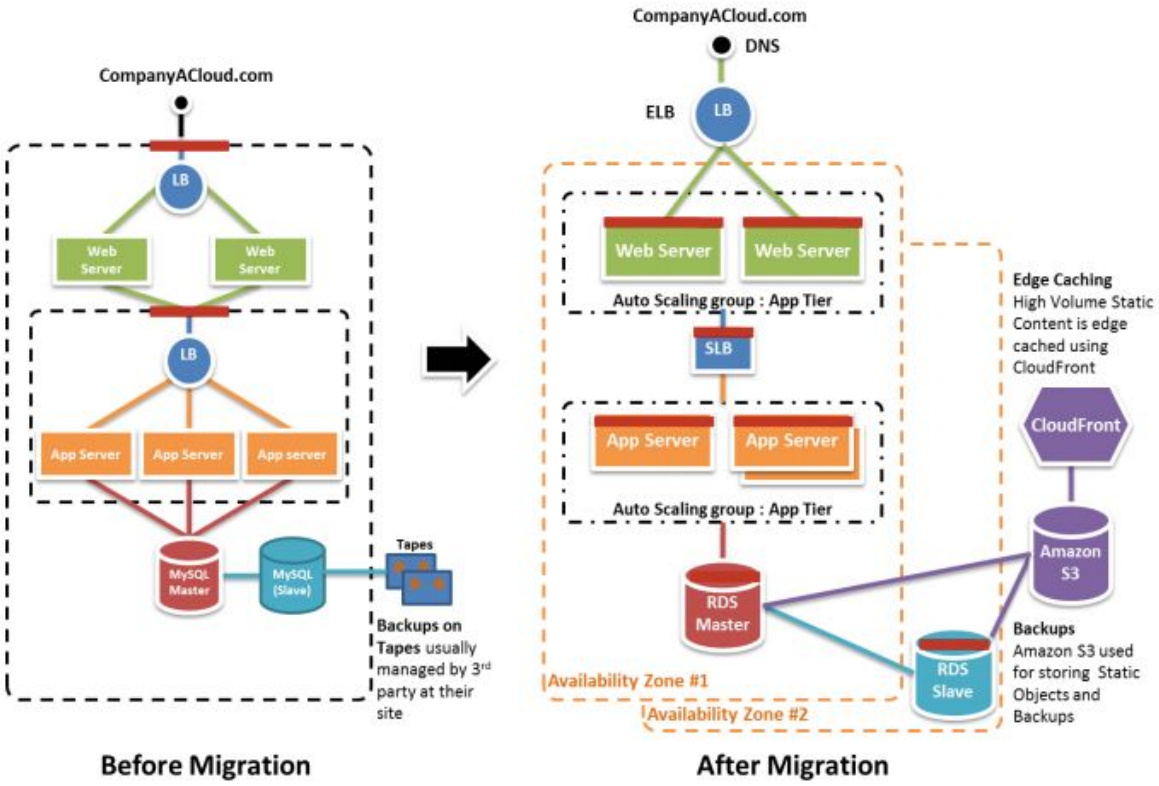


Figure 4: Minimum items to learn about services in a Proof of Concept

There are challenges of migrating the application deployment to AWS Cloud: (CONS)

1. No return on investment for the investment done in the current hardware
2. Limited applications available for interoperability of the systems on cloud. Hence, the flexibility is restricted as compared to full flexibility in case of on premise set up
3. Vendor lock in is a famous disadvantage evident in the cloud computing scenario. Sometimes if the on premise setup is deeply rooted in different modules of the legacy software, it becomes difficult for the migration of application on to the cloud
4. Additional functionality on cloud comes with additional costs

Architecture diagram before and after migration



References:

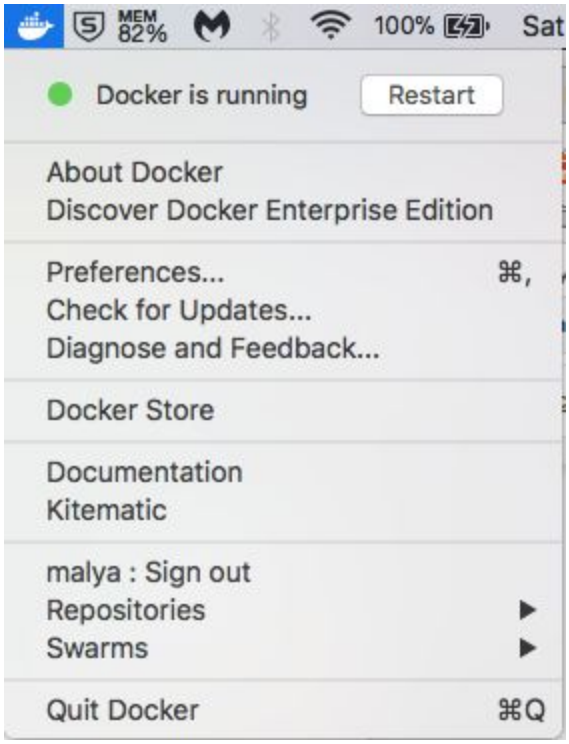
https://d1.awsstatic.com/whitepapers/compliance/AWS_HIPAA_Compliance_Whitepaper.pdf
https://aws.amazon.com/compliance/hipaa-compliance/
https://d0.awsstatic.com/whitepapers/cloud-migration-main.pdf

TECHNICAL PROJECT – SAME FOR BOTH THE GROUPS

- You create a **Stateless** Wordpress Docker Repository.
- Use this repository to deploy in the two Cloud vendors you are exploring.
- Ensure that the repository works out of the box on the two Cloud vendors within the Docker framework.
- Document the steps and show a working setup for both the Cloud vendors.
- Document why Docker is better with real measurements
- Which Cloud Vendor would you use and why

Creating stateless Wordpress Docker Repository

Installed Docker and updated its status to run



Verified the version of docker to be the latest

```
[MJ:~ Malya$ docker --version
Docker version 17.09.0-ce, build afdb6d4
```

Checking the first program of hello world to be running successfully on docker

```
[MJ:~ Malya$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

The next step shows creation of docker wordpress

```
[MJ:wordpress Malya$ docker run -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=wordpress --name wordpressdb -v "$
WD/database":/var/lib/mysql -d mariadb:latest database":/var/lib/mysql -d mariadb:latest
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
85b1f47fba49: Extracting [==================================================>  ]  51.25MB/52.6MB
5671503d4f93: Download complete
62466fedcc9d: Download complete
b4ef8399f3aa: Download complete
e34b2cf62e1d: Download complete
7291500bf826: Download complete
a77c97e1ff71: Download complete
5024f663c035: Download complete
dc69980fbc04: Download complete
e57f4562fa50: Download complete
49c749b145af: Download complete
```

Now installing the mysql database with mariaDB

```
[MJ:wordpress Malya$ docker run -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=wordpress --name wordpressdb -v "$
WD/database":/var/lib/mysql -d mariadb:latest database":/var/lib/mysql -d mariadb:latest
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
85b1f47fba49: Pull complete
5671503d4f93: Pull complete
62466fedcc9d: Pull complete
b4ef8399f3aa: Pull complete
e34b2cf62e1d: Pull complete
7291500bf826: Pull complete
a77c97e1ff71: Pull complete
5024f663c035: Pull complete
dc69980fbc04: Pull complete
e57f4562fa50: Pull complete
49c749b145af: Pull complete
Digest: sha256:338258203d1466202d2332e6f6860a3dd24d3bf3578c1e6999c847c975f7e4ca
Status: Downloaded newer image for mariadb:latest
6c8ebc7b943ae2775f4b920d70a6834cdd7801420f5953f13b00bc299dc96bde
MJ:wordpress Malya$
```

Checking the docker files contained - wordpressdb and mariadb

```
[MJ:wordpress Malya$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
        NAMES
6c8ebc7b943a        mariadb:latest      "docker-entrypoint..."   9 seconds ago       Up 11 seconds       3306/tcp
        wordpressdb
MJ:wordpress Malya$
```

Now using the pull command to download wordpress

```
[MJ:wordpress Malya$ docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
85b1f47fba49: Already exists
d8204bc92725: Extracting [=======>                                           ]   11.7MB/82.5MB
92fc16bb18e4: Download complete
31098e61b2ae: Download complete
f6ae64bfd33d: Download complete
003c1818b354: Download complete
a6fd4aeb32ad: Download complete
a094df7cedc1: Download complete
e3bf6fc1a51d: Download complete
ad235c260360: Download complete
edbf48bcbd7e: Download complete
fd6ae81d5745: Download complete
69838fd876d6: Download complete
3186ebffd72d: Download complete
b24a415ea2c0: Download complete
225bda14ea90: Download complete
fc0ad3550a92: Download complete
0e4600933a8c: Download complete
```

All pulls complete

```
[MJ:wordpress Malya$ docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
85b1f47fba49: Already exists
d8204bc92725: Pull complete
92fc16bb18e4: Pull complete
31098e61b2ae: Pull complete
f6ae64bfd33d: Pull complete
003c1818b354: Pull complete
a6fd4aeb32ad: Pull complete
a094df7cedc1: Pull complete
e3bf6fc1a51d: Pull complete
ad235c260360: Pull complete
edbf48bcbd7e: Pull complete
fd6ae81d5745: Pull complete
69838fd876d6: Pull complete
3186ebffd72d: Pull complete
b24a415ea2c0: Pull complete
225bda14ea90: Pull complete
fc0ad3550a92: Pull complete
0e4600933a8c: Pull complete
Digest: sha256:937862438b4f2a6b56193ef2cfd5fe345566e51278be56a04a7dfcdde18c5922
Status: Downloaded newer image for wordpress:latest
MJ:wordpress Malya$
```

Modifying the docker-compose.yaml file

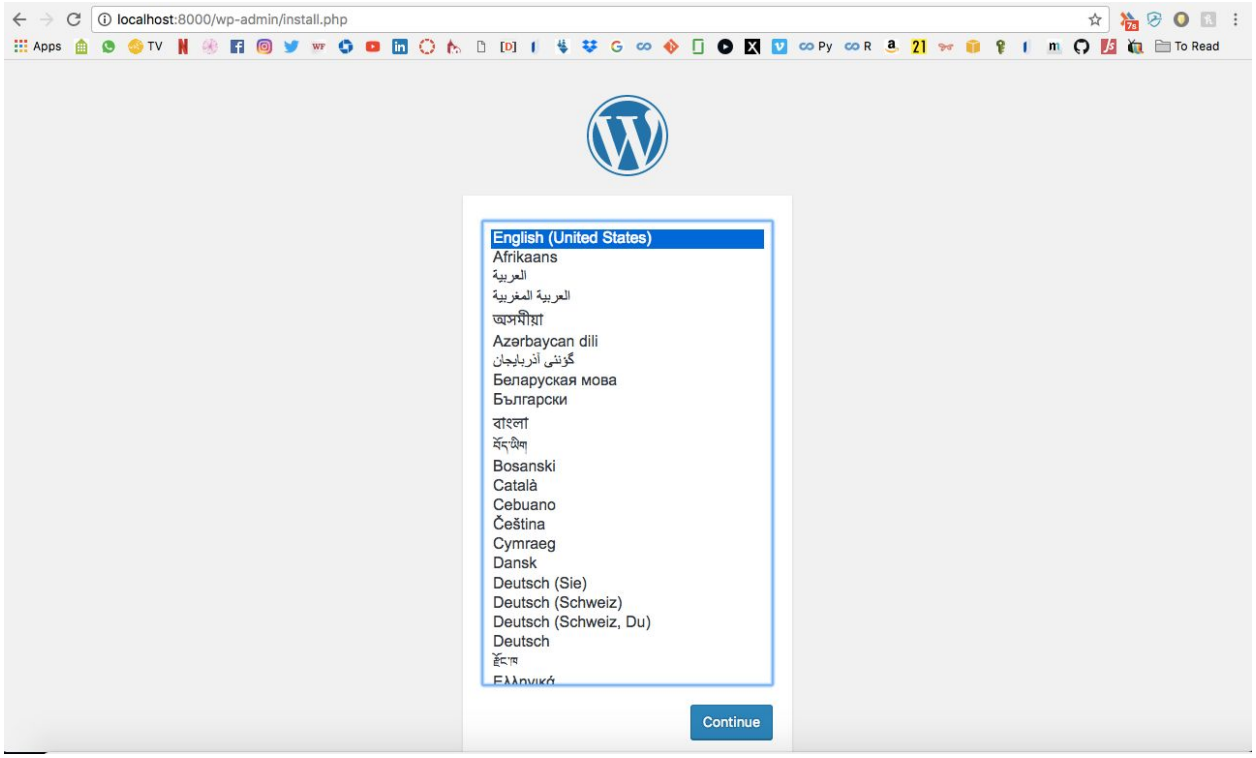
```
version: '3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: malyajain
       MYSQL_DATABASE: malyajain
       MYSQL_USER: malyajain
       MYSQL_PASSWORD: malyajain

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
~
~
~
~
```

Running the docker file

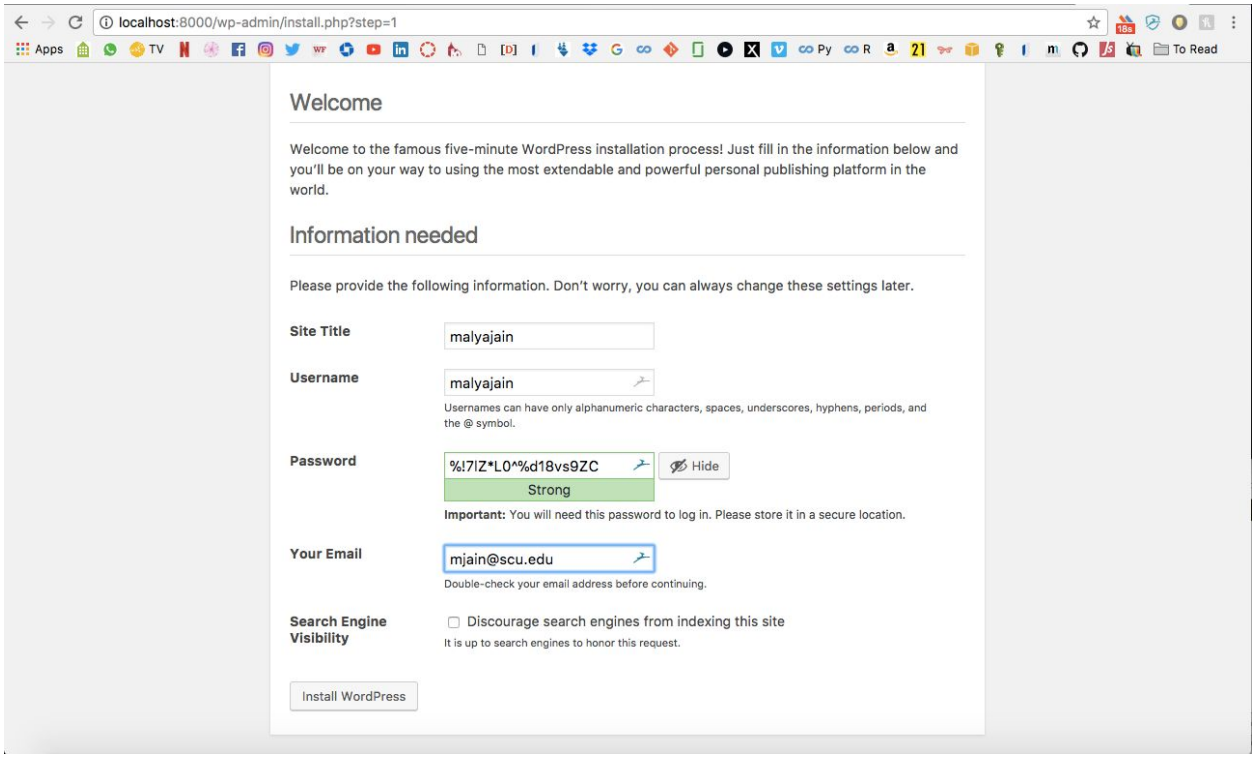
```
[MJ:~ Malya$ sudo vi docker-compose.yml
[Password:
[MJ:~ Malya$ docker run -e WORDPRESS_DB_PASSWORD=malyajain --name malyajain --link wordpressdb:mysql -p 127.0.0.1:80:]
80 -v "$PWD/html":/var/www/html -d wordpress
3c8c65ffba72501cad1b97c1e0c97747eb98de1c99fbdccef76efb12a621c2c8
MJ:~ Malya$
```

```
MJ:~ Malya$ docker-compose up -d
Creating network "malya_default" with the default driver
Creating volume "malya_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
85b1f47fba49: Already exists
5671503d4f93: Already exists
3b43b3b913cb: Pull complete
4fbb803665d0: Pull complete
05808866e6f9: Pull complete
1d8c65d48cfa: Pull complete
e189e187b2b5: Pull complete
02d3e6011ee8: Extracting [==============>                                    ]  22.84MB/79.56MB
d43b32d5ce04: Download complete
2a809168ab45: Download complete
```

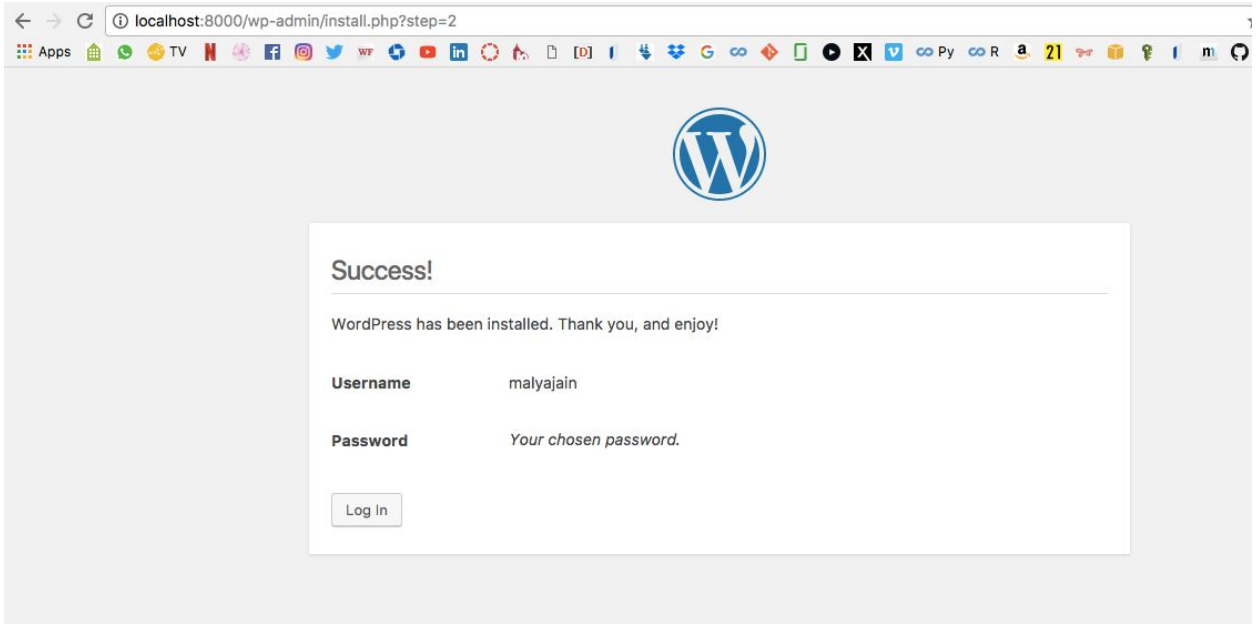
Opening wordpress locally

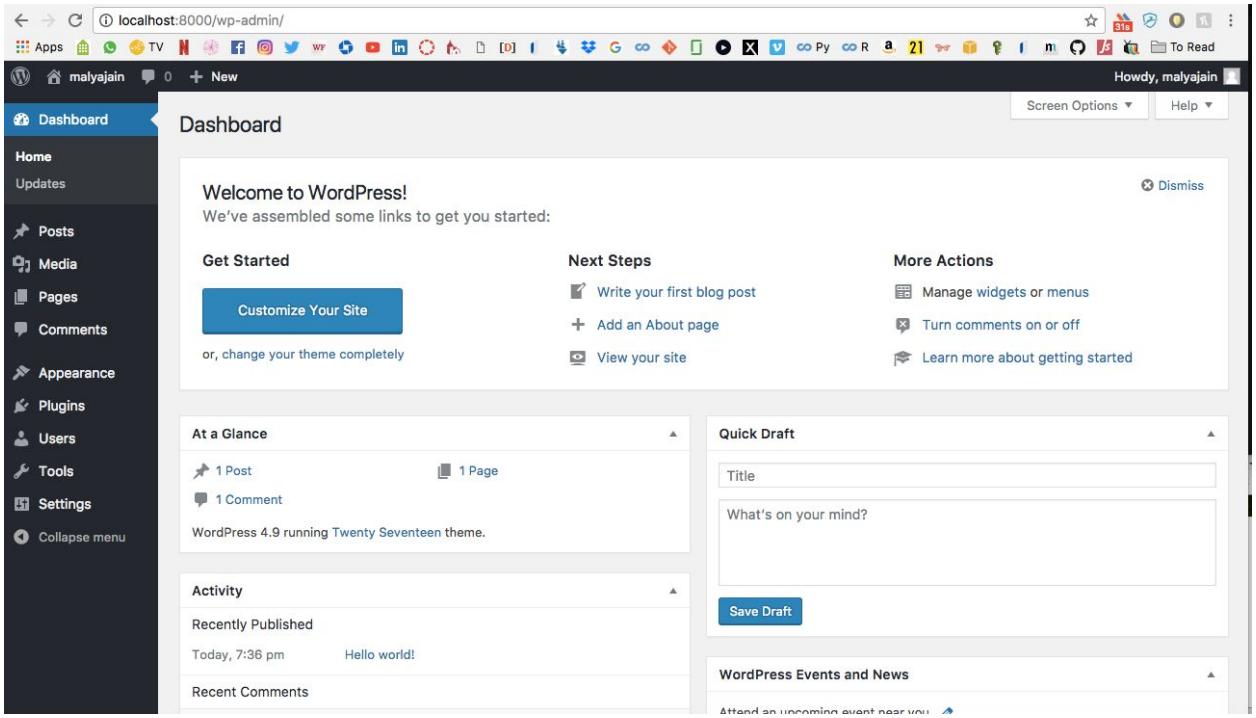


Creating the login information



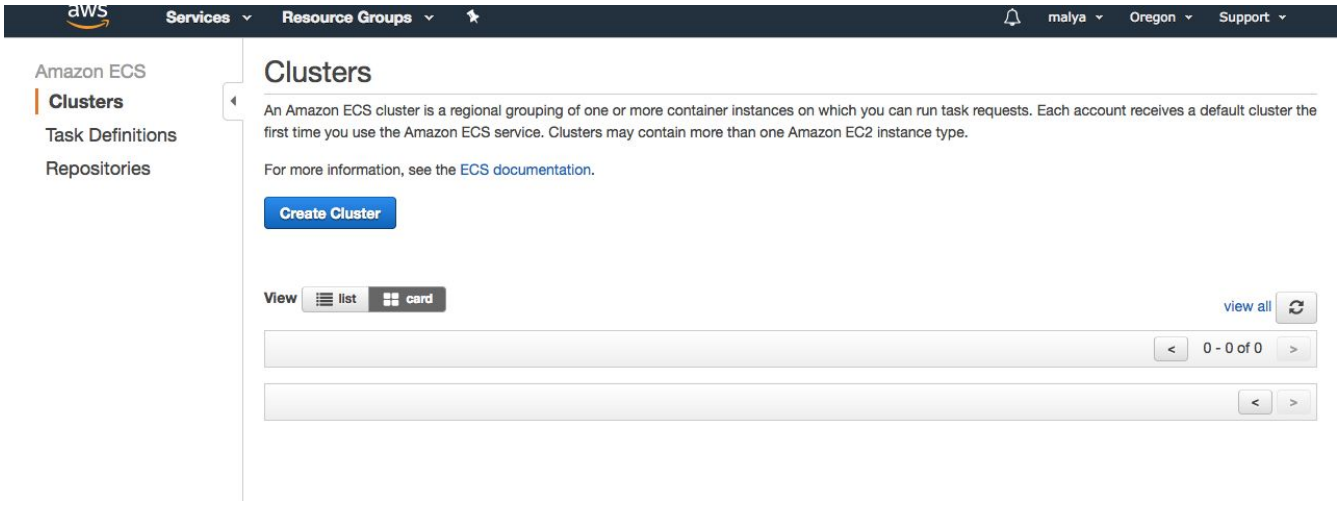
Wordpress installed on the docker locally

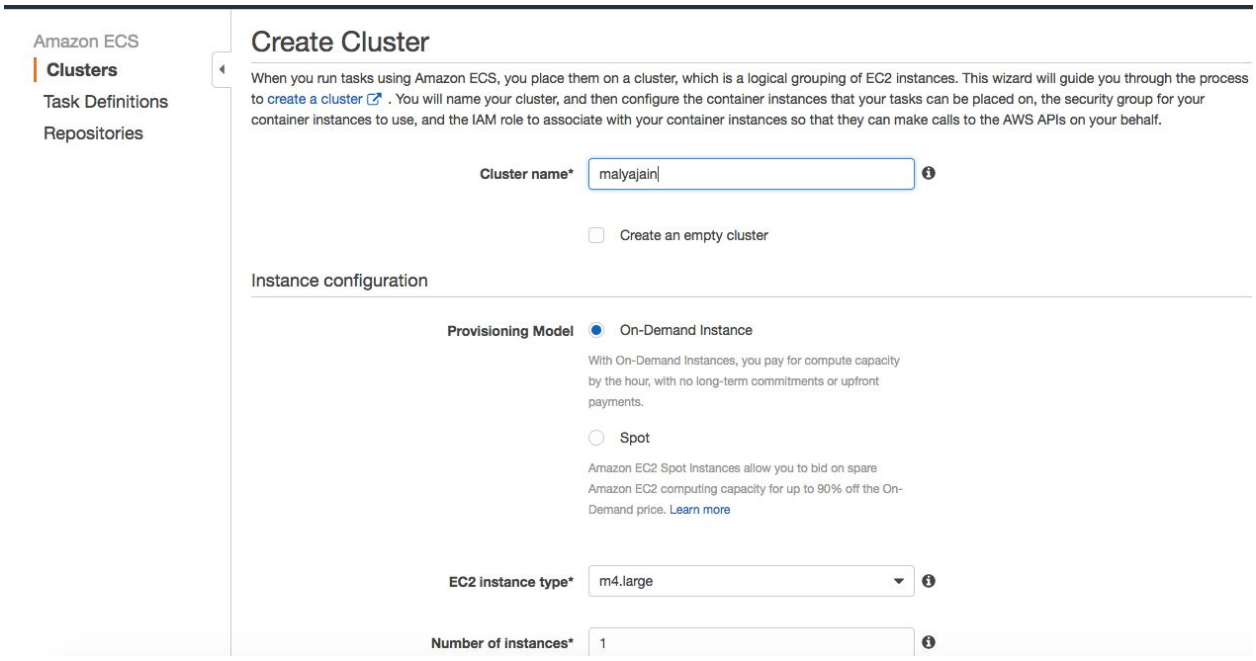


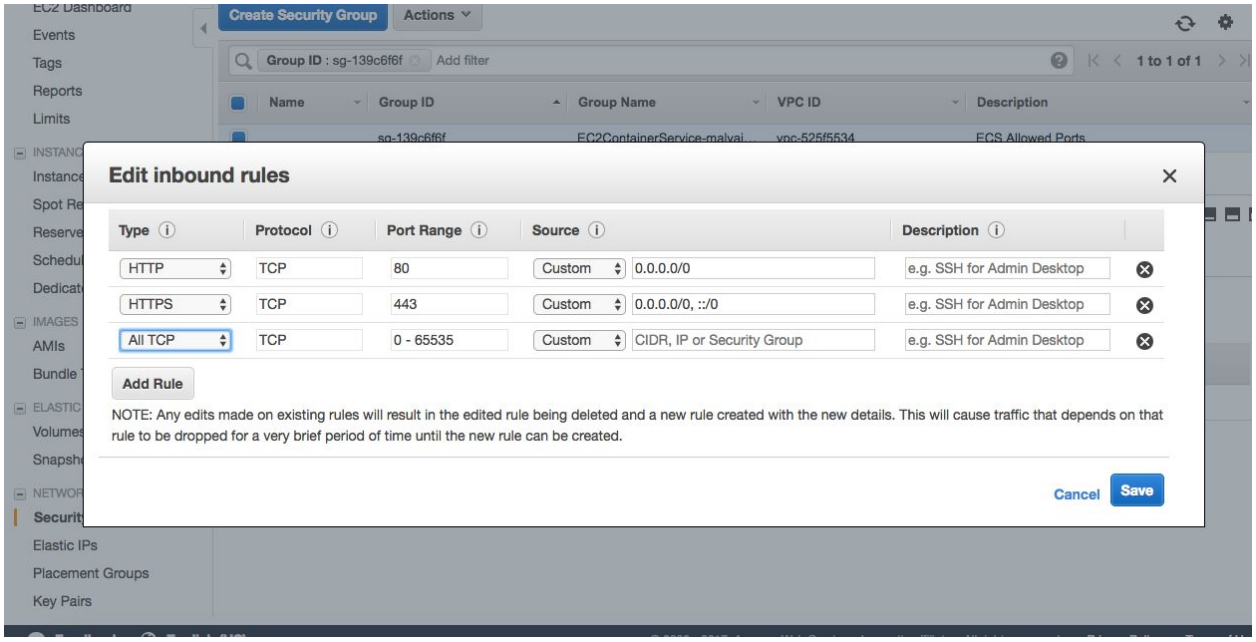


AWS URL: http://52.37.163.173

Deploying the wordpress via Docker on ECS




Amazon ECS
Clusters
Task Definitions
Repositories
Create Cluster
When you run tasks using Amazon ECS, you place them on a cluster, which is a logical grouping of EC2 instances. This wizard will guide you through the process to create a cluster . You will name your cluster, and then configure the container instances that your tasks can be placed on, the security group for your container instances to use, and the IAM role to associate with your container instances so that they can make calls to the AWS APIs on your behalf.
Cluster name*
malyajain
Create an empty cluster
Instance configuration
Provisioning Model
On-Demand Instance
With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.
Spot
Amazon EC2 Spot Instances allow you to bid on spare Amazon EC2 computing capacity for up to 90% off the On-Demand price. Learn more
EC2 instance type*
m4.large
Number of instances*
1


Create Security Group
Actions
Group ID : sg-139c6f6f
Add filter
1 to 1 of 1
Name
Group ID
Group Name
VPC ID
Description
Edit inbound rules
Type
Protocol
Port Range
Source
Description
HTTP
TCP
80
Custom
0.0.0.0/0
e.g. SSH for Admin Desktop
HTTPS
TCP
443
Custom
0.0.0.0/0, ::/0
e.g. SSH for Admin Desktop
All TCP
TCP
0 - 65535
Custom
CIDR, IP or Security Group
e.g. SSH for Admin Desktop
Add Rule
NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.
Cancel
Save
Events
Tags
Reports
Limits
Elastic IPs
Placement Groups
Key Pairs

ECS status - 3 of 3 complete **malyajain**

ECS cluster
ECS Cluster malyajain successfully created

ECS Instance IAM Policy
IAM Policy for the role ecsInstanceRole successfully attached

CloudFormation Stack
CloudFormation stack EC2ContainerService-malyajain and its resources successfully created

Cluster Resources

Instance type	m4.large
Desired number of instances	1
Key pair	Oct07
ECS AMI ID	ami-3702ca4f
VPC	vpc-d2515bb4
Subnet 1	subnet-288efe60
Subnet 1 route table association	rtbassoc-ec28e497
Subnet 2	subnet-99bfabc2
Subnet 2 route table association	rtbassoc-af2ce0d4
VPC Availability Zones	us-west-2b, us-west-2c, us-west-2a
Security group	sg-55807329
Internet gateway	igw-c02e3da7
Route table	rtb-93a1f4ea
Amazon EC2 route	EC2Co-Publi-AEL3LHNBBU3M
Virtual private gateway attachment	EC2Co-Attac-14SYHUH1M3F2L

Clusters › malyajain

Cluster : malyajain

Delete Cluster

Get a detailed view of the resources on your cluster.

Status	ACTIVE
Registered container instances	1
Pending tasks count	0
Running tasks count	0

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks

Create | Update | Delete

Last updated on November 25, 2017 12:08:34 PM (0m ago)

Filter in this page

Service Name	Status	Task Definition	Desired tasks	Running tasks
No results				

Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you can run task requests. Each account receives a default cluster the first time you use the Amazon ECS service. Clusters may contain more than one Amazon EC2 instance type.

For more information, see the ECS documentation.

Create Cluster

View: list | card

view all

< 1 - 1 of 1 >

malyajain >

0	0	0	0.00%	0.00%	1
Services	Running tasks	Pending tasks	CPUUtilization	MemoryUtilization	Container instances

< >

Secure | https://us-west-2.console.aws.amazon.com/ecs/home?region=us-west-2#/clusters/malyajain/containerInstances

Apps TV To Read

aws | Services | Resource Groups | malya | Oregon | Support

Amazon ECS
- **Clusters**
- Task Definitions
- Repositories

Clusters > malyajain

Cluster : malyajain

Delete Cluster

Get a detailed view of the resources on your cluster.

Status	ACTIVE
Registered container instances	1
Pending tasks count	0
Running tasks count	0

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks

Scale ECS Instances | Actions | Last updated on November 25, 2017 12:09:15 PM (0m ago)

Status: ALL ACTIVE DRAINING | 1-1 | Page size 50

Filter by attributes (click or press down arrow to view filter options)

Container Instance	EC2 Instance	Availability Zo...	Agent Connec...	Status	Running tasks...	CPU availab
6a67a937-9763-416b-80ef...	i-08ba5614e39f...	us-west-2c	true	ACTIVE	0	2048

Clusters > malyajain > Container Instance: 6a67a937-9763-416b-80ef-7d1ab5995784

Container Instance : 6a67a937-9763-416b-80ef-7d1ab5995784

Update agent | Deregister

Details

Cluster	malyajain
EC2 Instance	i-08ba5614e39fb5ae1
Availability Zone	us-west-2c
Public DNS	ec2-52-24-214-43.us-west-2.compute.amazonaws.com
Private DNS	ip-10-0-1-93.us-west-2.compute.internal
Public IP	52.24.214.43
Private IP	10.0.1.93
Status	ACTIVE
Running tasks count	0
Agent version	1.15.0
Docker version	17.06.2-ce

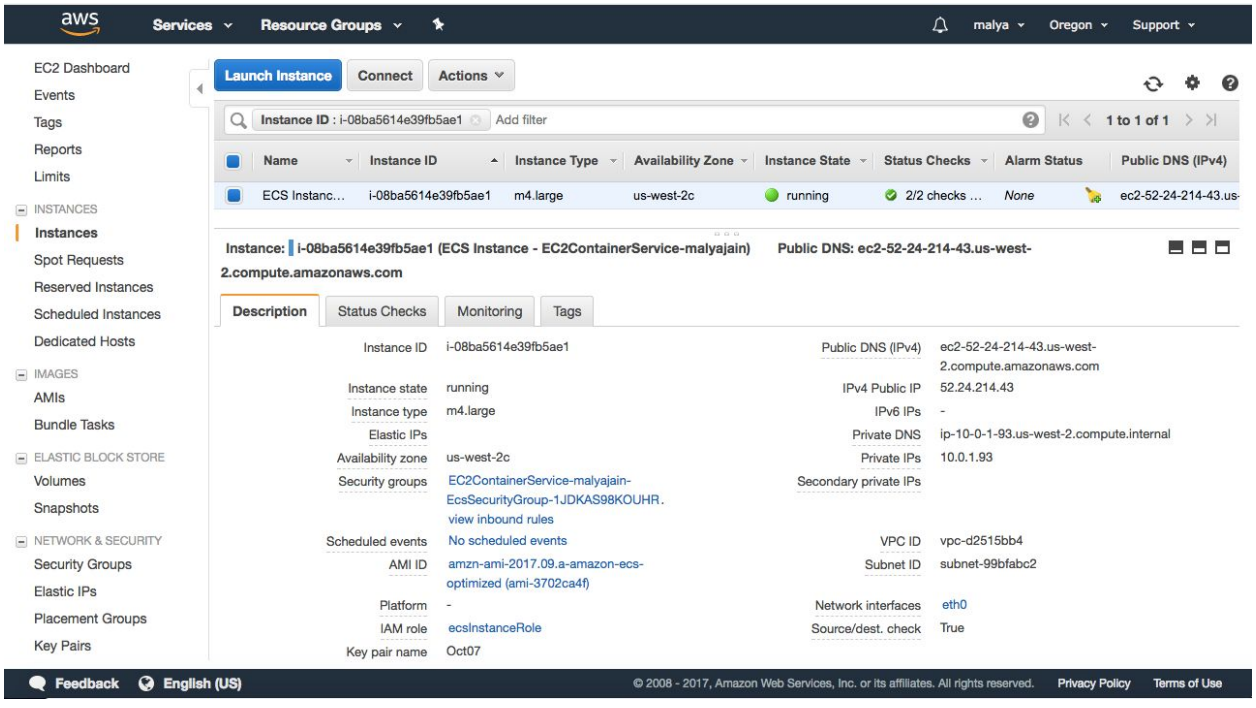
Resources

Resources	Registered	Available
CPU	2048	2048
Memory	7985	7985
Ports	*5 ports*	

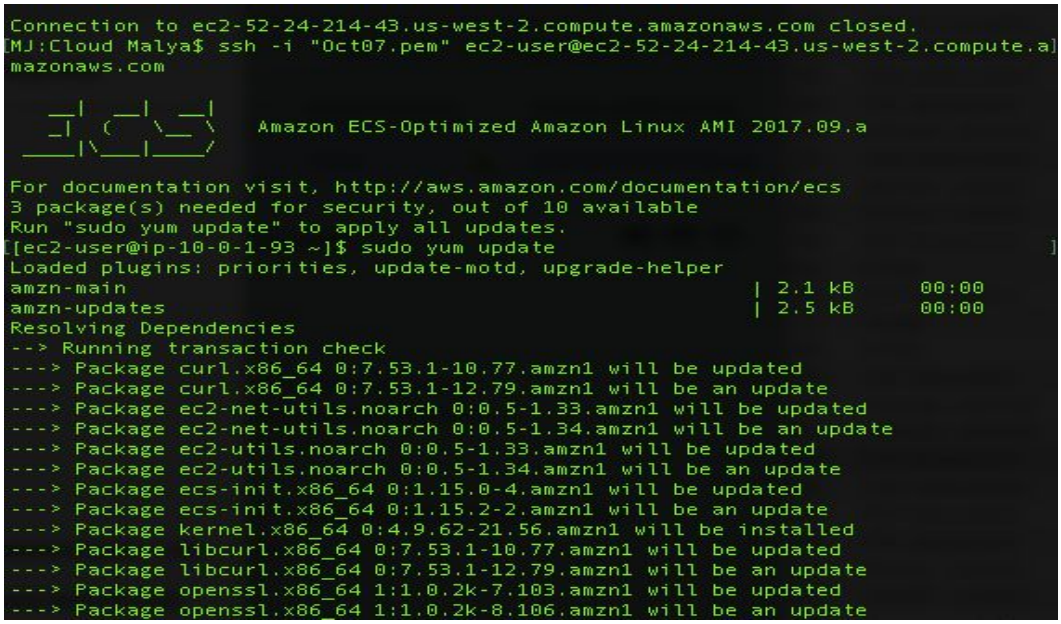
Tasks

Stop | Stop All

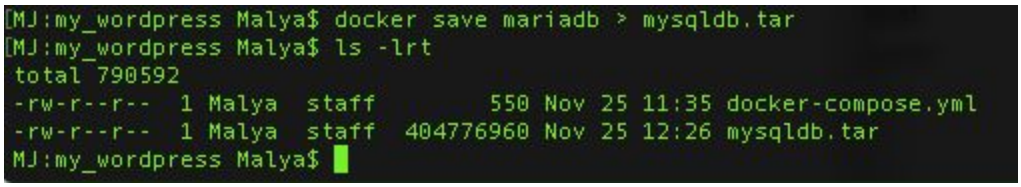
Last updated on November 25, 2017 12:09:24 PM (0m ago)



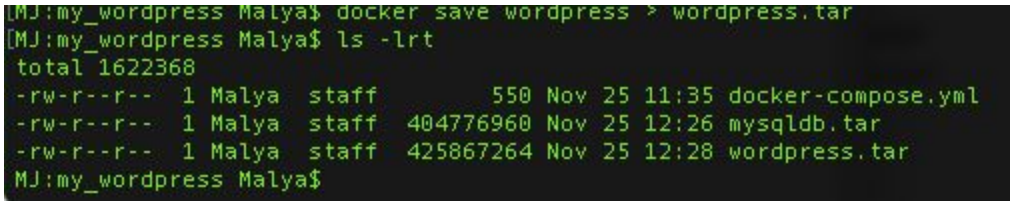
SSH-ing into the instance



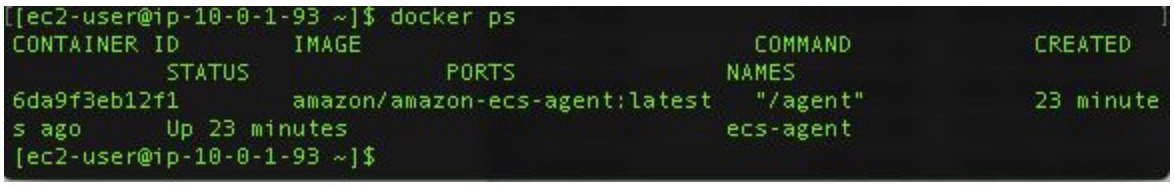
Saving maria-db in docker



Saving wordpress in docker



Docker files



SCP into the instance for wordpress.tar and mysqld.tar

```
[MJ:Cloud Malya$ scp -i "Oct07.pem" /Users/Malya/my_wordpress/wordpress.tar ec2-user@ec2-52-24-214-43.us-west-2.]
compute.amazonaws.com:
wordpress.tar                                                 100%  406MB   1.4MB/s   04:48
[MJ:Cloud Malya$ scp -i "Oct07.pem" /Users/Malya/my_wordpress/mysqldb.tar ec2-user@ec2-52-24-214-43.us-west-2.co]
mpute.amazonaws.com:
mysqldb.tar                                                   100%  386MB   1.4MB/s   04:33
MJ:Cloud Malya$
```

SSH into the instance again

```
Last login: Sat Nov 25 15:58:25 on ttys001
[MJ:~ Malya$ cd Desktop/Cloud/
MJ:Cloud Malya$ chmod 400 MyKeyPair.pem
[MJ:Cloud Malya$ ssh -i "MyKeyPair.pem" ec2-user@ec2-52-35-50-42.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-52-35-50-42.us-west-2.compute.amazonaws.com (52.35.50.42)' can't be established.
ECDSA key fingerprint is SHA256:put0E9WZwExykxutTbIeb+QjfS/llic4nqT14hzer8g.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-35-50-42.us-west-2.compute.amazonaws.com,52.35.50.42' (ECDSA) to the list of known hosts.

   __|  __|  __|
   _|  (   \__ \   Amazon ECS-Optimized Amazon Linux AMI 2017.09.a
 ____|\___|____/

For documentation visit, http://aws.amazon.com/documentation/ecs
3 package(s) needed for security, out of 10 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-1-197 ~]$ sudo yum install
Loaded plugins: priorities, update-motd, upgrade-helper
Error: Need to pass a list of pkgs to install
```

Loading the docker files into AWS instance

```
For documentation visit, http://aws.amazon.com/documentation/ecs
[[ec2-user@ip-10-0-1-93 ~]$ ls
home  mysqldb.tar  wordpress.tar
[[ec2-user@ip-10-0-1-93 ~]$ docker load < wordpress.tar
c01c63c6823d: Loading layer [==================================================>]  129.3MB/129.3MB
37412c153883: Loading layer [==================================================>]  204.7MB/204.7MB
c3d26400d3ff: Loading layer [==================================================>]  3.584kB/3.584kB
5cd2e0cfe892: Loading layer [==================================================>]  8.552MB/8.552MB
2c3aa4e96952: Loading layer [==================================================>]  10.24kB/10.24kB
4c0354ed71f4: Loading layer [==================================================>]  9.728kB/9.728kB
2f6273a5f133: Loading layer [==================================================>]  4.096kB/4.096kB
dcdbe9fe2ca1: Loading layer [==================================================>]   7.68kB/7.68kB
a9aa8861270e: Loading layer [==================================================>]  13.87MB/13.87MB
fa7f9311a060: Loading layer [==================================================>]  4.096kB/4.096kB
61a961ab5d2b: Loading layer [==================================================>]  33.01MB/33.01MB
8933dc910eee: Loading layer [==================================================>]  11.78kB/11.78kB
493137409f3e: Loading layer [==================================================>]  4.608kB/4.608kB
749e8aaa7dd4: Loading layer [==================================================>]  6.892MB/6.892MB
6594bf4ea5b9: Loading layer [==================================================>]  4.608kB/4.608kB
24605e7ca88b: Loading layer [==================================================>]  7.168kB/7.168kB
8911d4754681: Loading layer [==================================================>]  29.41MB/29.41MB
d47b5306d1bf: Loading layer [==================================================>]  10.24kB/10.24kB
Loaded image: wordpress:latest
```

```
[ec2-user@ip-10-0-1-93 ~]$ docker load < mysqldb.tar
10d8a8f4f236: Loading layer [==================================================>]  345.1kB/345.1kB
dbdf99ad9655: Loading layer [==================================================>]  3.178MB/3.178MB
34974b99e70a: Loading layer [==================================================>]  1.536kB/1.536kB
3fd7d06644cf: Loading layer [==================================================>]  15.05MB/15.05MB
98c17e47ef10: Loading layer [==================================================>]   25.6kB/25.6kB
501f5926a230: Loading layer [==================================================>]   5.12kB/5.12kB
f0100bc536f0: Loading layer [==================================================>]   5.12kB/5.12kB
44c95ff56282: Loading layer [==================================================>]  256.9MB/256.9MB
e0e224ab5dbd: Loading layer [==================================================>]  8.704kB/8.704kB
0cea6c4a3072: Loading layer [==================================================>]  1.536kB/1.536kB
```

```
[[ec2-user@ip-10-0-1-93 ~]$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
wordpress                   latest              467f492bc127        4 days ago          413MB
mariadb                     latest              abcee1d29aac        9 days ago          396MB
amazon/amazon-ecs-agent     latest              431c8544194f        9 days ago          25.5MB
amazon/amazon-ecs-pause     0.1.0               2980f5c2a885        9 days ago          964kB
<none>                      <none>              003a2d4911ad        3 weeks ago         25.5MB
<none>                      <none>              7e9d1d0c6b74        3 weeks ago         964kB
```

Running the docker repository on instance

```
[[root@ip-10-0-1-93 ec2-user]# docker run -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=wordpress --name wor]
dpressdb -v "$PWD/database":/var/lib/mysql -d mariadb:latest
876807b7f7672e5f3ee6246d4d896b79e68e4c8ea662f145e1d739ad0036526a
```

```
[root@ip-10-0-1-93 ec2-user]# docker run -e WORDPRESS_DB_PASSWORD=test --name wordpress --link wordpressdb:mysq]
l -p 127.0.0.1:80:80 -v "$PWD/html":/var/www/html -d wordpress
06897be7ef20cc344f66a4098d7b4721d1f533b4b2604c2b0561457daccad537
```

```
[root@ip-10-0-1-93 ec2-user]# docker run -e WORDPRESS_DB_PASSWORD=mallyajain --name wordpressdocker --link wordpressdb:mysql -]
p 127.0.0.1:80:80 -v "$PWD/html":/var/www/html -d wordpress
57ac424e49f2ce3f16b8b1af3ab8a409601bafcb660054243f43770c5e793861
```

```
[root@ip-10-0-1-93 ec2-user]# docker run -e WORDPRESS_DB_PASSWORD=password -d --name wordpress --link wordpressdb:mysql -p 127.0.0.2:8080:80 -v "$PWD/":/var/www/ht]
ml  wordpress
974fddb511c4a556bb9aae2795e4b2454784831b0a09c73642a2ac7ce8a8bb20
```

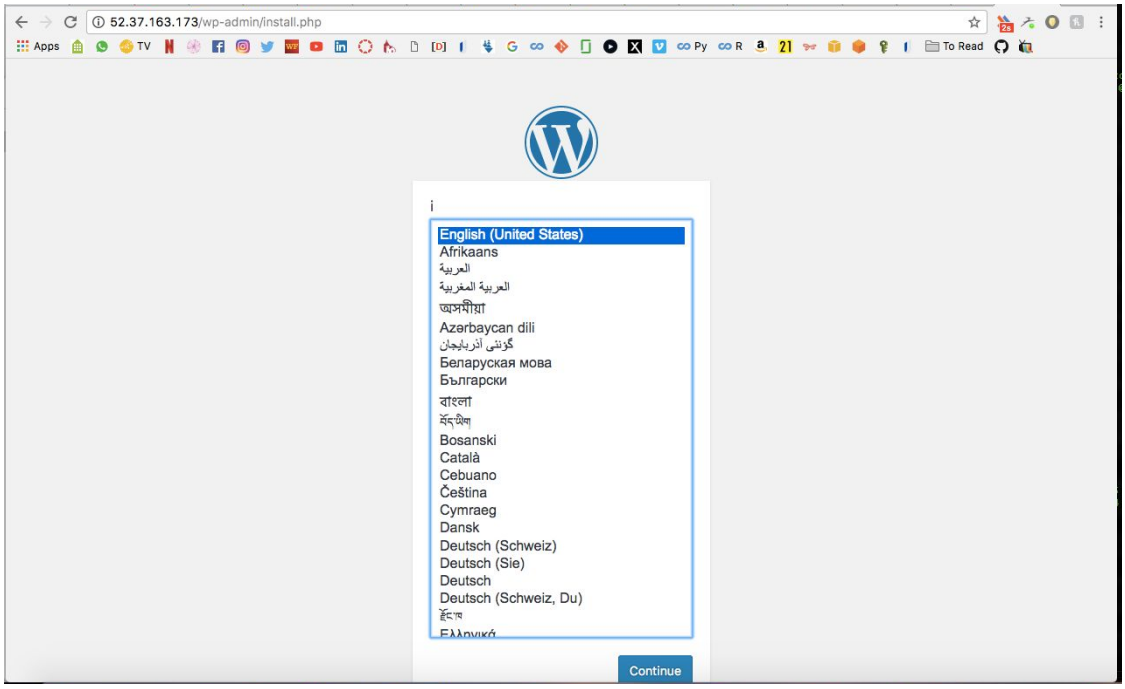
Running images

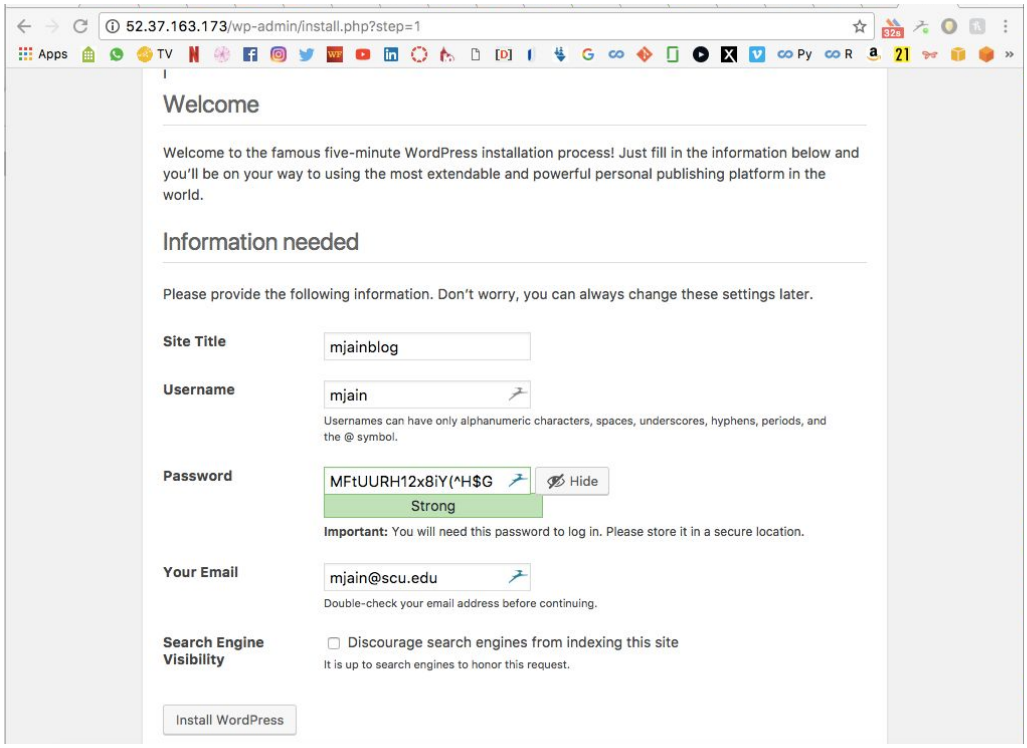
```
[root@ip-10-0-1-93 ec2-user]# docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                    NAMES
974fddb511c4        wordpress                      "docker-entrypoint..."   14 seconds ago      Up 13 seconds       127.0.0.2:8080->80/tcp   wordpress
876807b7f767        mariadb:latest                 "docker-entrypoint..."   2 hours ago         Up 2 hours          3306/tcp                 wordpressdb
6da9f3eb12f1        amazon/amazon-ecs-agent:latest "/agent"                  3 hours ago         Up 3 hours                                   ecs-agent
[root@ip-10-0-1-93 ec2-user]#
```

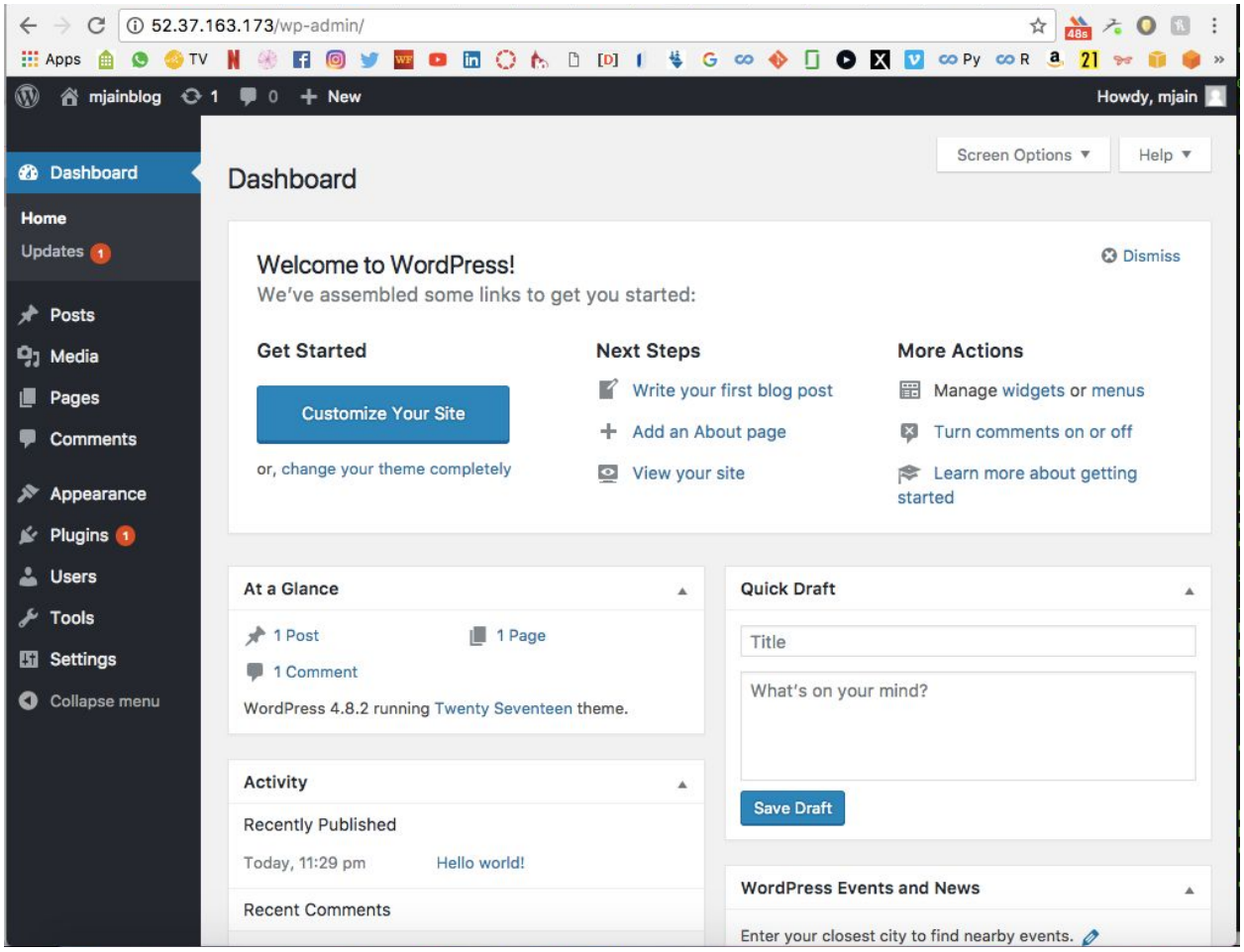
Pulling the registry

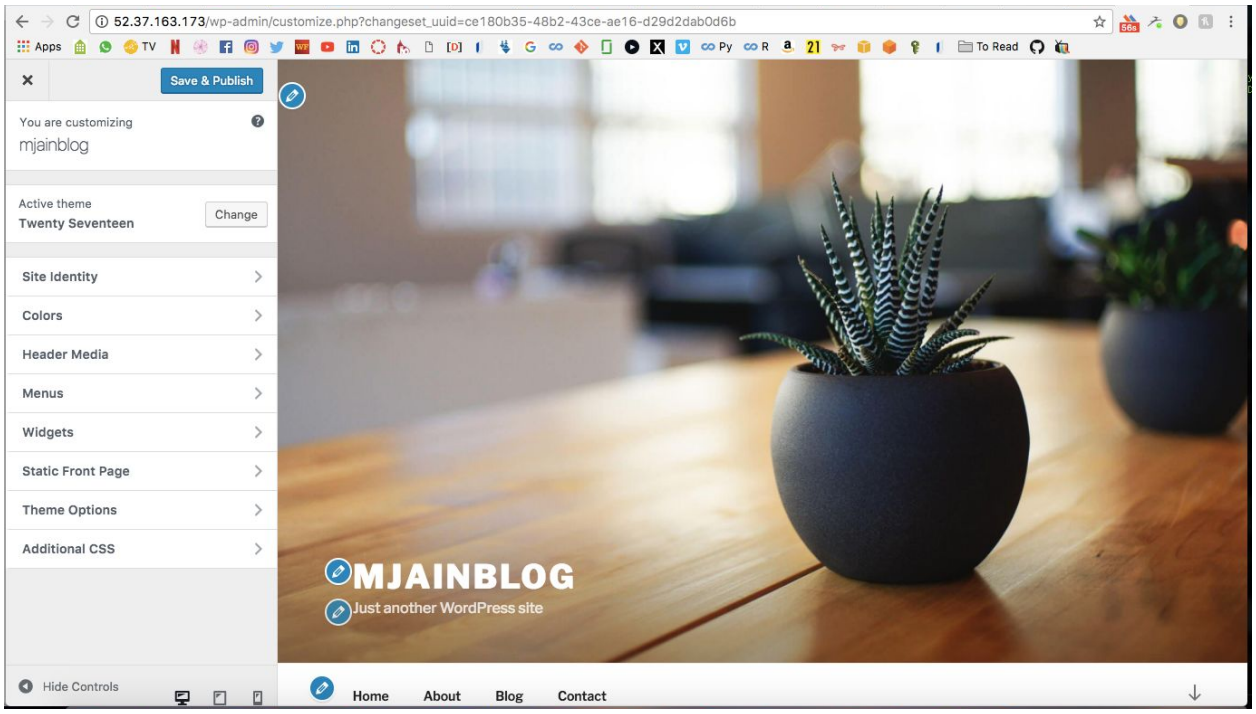
```
[root@ip-10-0-1-93 ec2-user]# docker pull registry
Using default tag: latest
latest: Pulling from library/registry
49388a8c9c86: Pull complete
e4d43608dd22: Pull complete
3a41740f900c: Pull complete
e16ef4b76684: Pull complete
65f212f7c778: Pull complete
Digest: sha256:d837de65fd9bdb81d74055f1dc9cc9154ad5d8d5328f42f57f273000c402c76d
Status: Downloaded newer image for registry:latest
[root@ip-10-0-1-93 ec2-user]#
```

Installing the Wordpress on EC2 via docker



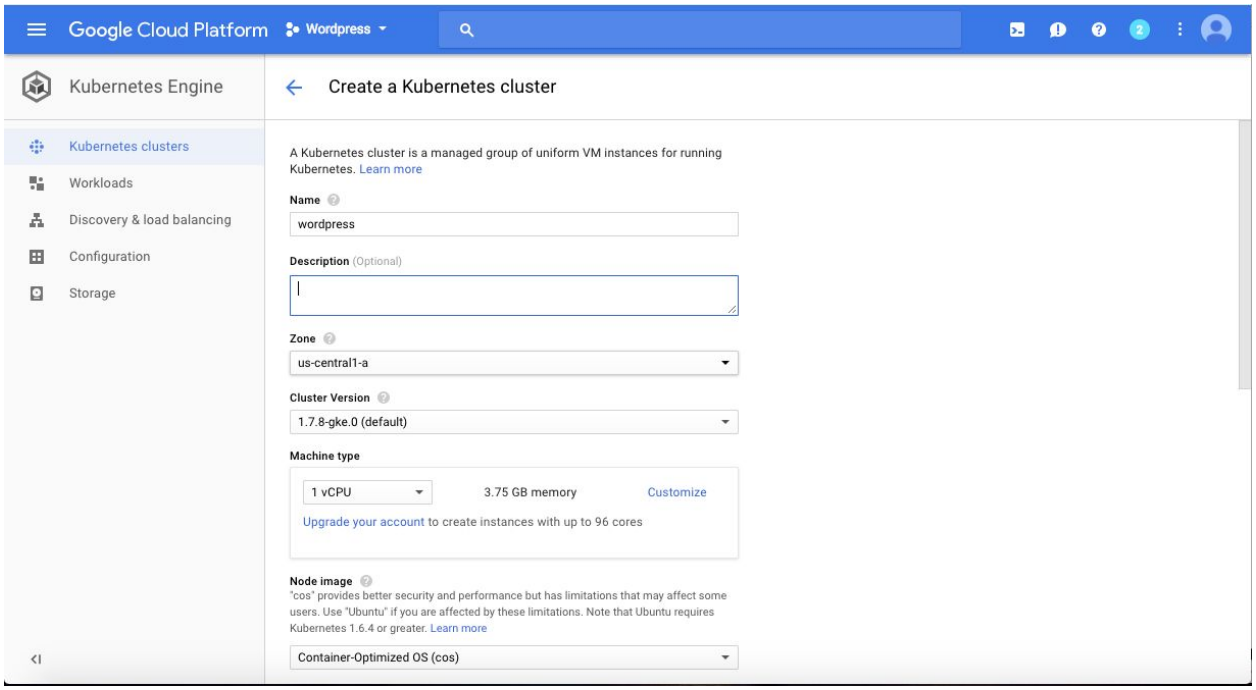



52.37.163.173/wp-admin/
mjainblog
1
0
New
Howdy, mjain
Dashboard
Home
Updates 1
Posts
Media
Pages
Comments
Appearance
Plugins 1
Users
Tools
Settings
Collapse menu
Screen Options
Help
Dashboard
Welcome to WordPress!
We've assembled some links to get you started:
Dismiss
Get Started
Customize Your Site
or, change your theme completely
Next Steps
Write your first blog post
Add an About page
View your site
More Actions
Manage widgets or menus
Turn comments on or off
Learn more about getting started
At a Glance
1 Post
1 Page
1 Comment
WordPress 4.8.2 running Twenty Seventeen theme.
Activity
Recently Published
Today, 11:29 pm
Hello world!
Recent Comments
Quick Draft
Title
What's on your mind?
Save Draft
WordPress Events and News
Enter your closest city to find nearby events.

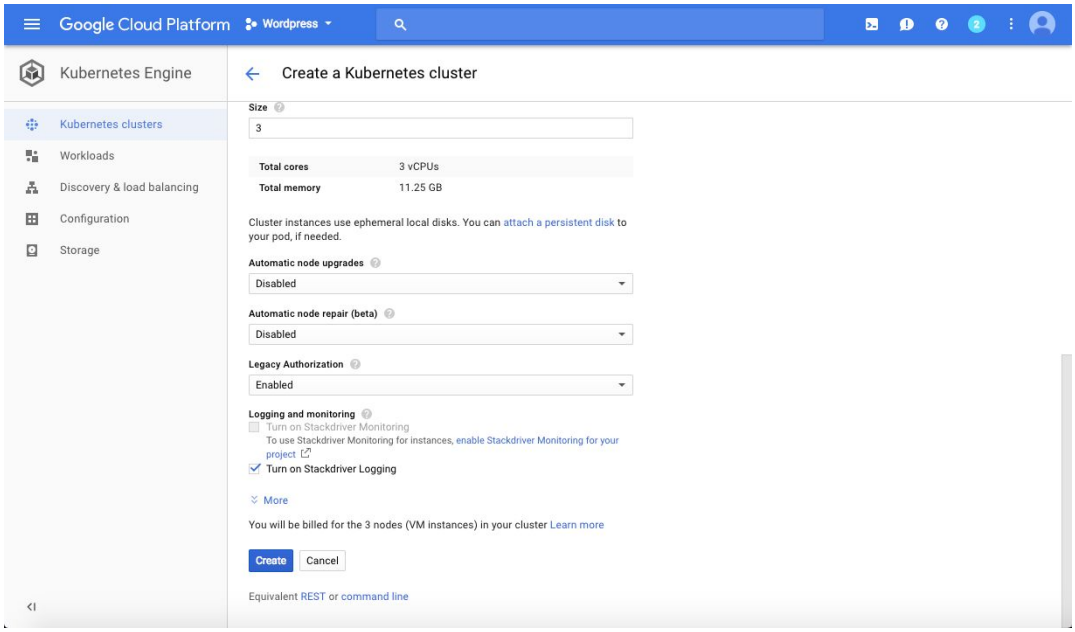

52.37.163.173/wp-admin/customize.php?changeset_uuid=ce180b35-48b2-43ce-ae16-d29d2dab0d6b
Save & Publish
You are customizing
mjainblog
Active theme
Twenty Seventeen
Change
Site Identity
Colors
Header Media
Menus
Widgets
Static Front Page
Theme Options
Additional CSS
Hide Controls
MJAINBLOG
Just another WordPress site
Home
About
Blog
Contact

Google Cloud platform URL http://104.154.207.249.xip.io/

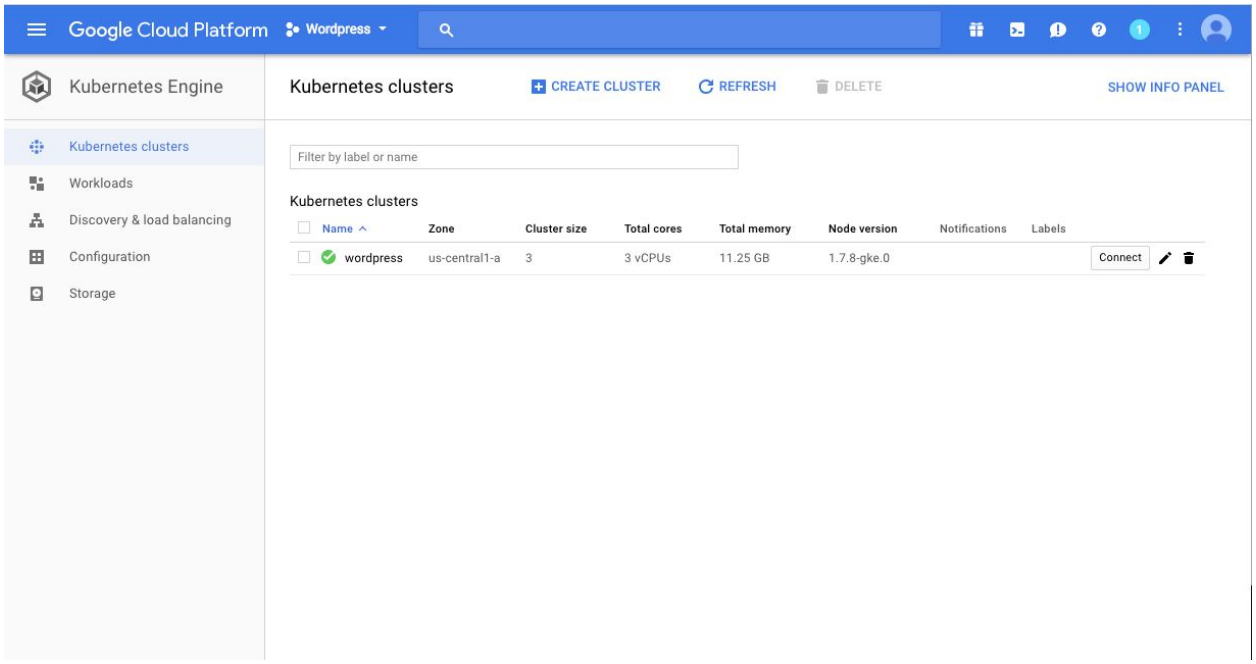
Installed the Docker on Kubernetes container services
Created a new project called Wordpress and a Kubernetes container called wordpress



Defined to have 3 nodes on the cluster



The Kubernetes cluster is up and running



Pulling the local docker wordpress file

```
docker: Error response from daemon: repository malyajain/wordpress not found: does not exist or no pull access.
See 'docker run --help'.
malyajain@wordpress-187204:~$ docker run -i -d -p 80:80 wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
85b1f47fba49: Pulling fs layer
d8204bc92725: Pulling fs layer
92fc16bb18e4: Pulling fs layer
31098e61b2ae: Pulling fs layer
d8204bc92725: Extracting [==========================================>    ]  77.43MB/82.5MB
003c1818b354: Download complete
a6fd4aeb32ad: Download complete
a094df7cedc1: Download complete
e3bf6fc1a51d: Download complete
ad235c260360: Download complete
```

```
malyajain@wordpress-187204:~$ docker run -i -d -p 80:80 wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
85b1f47fba49: Pulling fs layer
d8204bc92725: Pulling fs layer
92fc16bb18e4: Pulling fs layer
31098e61b2ae: Pulling fs layer
d8204bc92725: Extracting [==================>                            ]  33.42MB/82.5MB
003c1818b354: Download complete
a6fd4aeb32ad: Download complete
a094df7cedc1: Download complete
e3bf6fc1a51d: Download complete
```

The pull is complete

```
malyajain@wordpress-187204:~$ docker run -i -d -p 80:80 wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
85b1f47fba49: Pulling fs layer
d8204bc92725: Pulling fs layer
92fc16bb18e4: Pulling fs layer
31098e61b2ae: Pulling fs layer
f6ae64bfd33d: Pull complete
003c1818b354: Pull complete
a6fd4aeb32ad: Pull complete
a094df7cedc1: Pull complete
e3bf6fc1a51d: Pull complete
ad235c260360: Pull complete
edbf48bcbd7e: Pull complete
fd6ae81d5745: Pull complete
69838fd876d6: Pull complete
3186ebffd72d: Pull complete
b24a415ea2c0: Pull complete
225bda14ea90: Pull complete
fc0ad3550a92: Pull complete
0e4600933a8c: Pull complete
Digest: sha256:5b3b36db3c19d5b8c6ded6facec4daac57fe2ea1879351a2e65ac8919cea37ce
Status: Downloaded newer image for wordpress:latest
739f74462e7dfe6c1949bf193ef5ba3717afdfee7e802c4f1165f91e9c65592b
docker: Error response from daemon: driver failed programming external connectivity on endpoint zealous_lumiere (ebb4f6597a067e3b5cdb02c140def0e924296781396f6b52e337caf9
b76b5185): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: address already in use.
```

Pulling the docker image for the mySQL latest

```
malyajain@wordpress-187204:~$ docker run -i -d -p 80:80 mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
85b1f47fba49: Already exists
5671503d4f93: Pull complete
3b43b3b913cb: Pull complete
4fbb803665d0: Pull complete
05808866e6f9: Pull complete
1d8c65d48cfa: Pull complete
e189e187b2b5: Pull complete
02d3e6011ee8: Extracting [==================================================>]  79.56MB/79.56MB
d43b32d5ce04: Download complete
2a809168ab45: Download complete
```

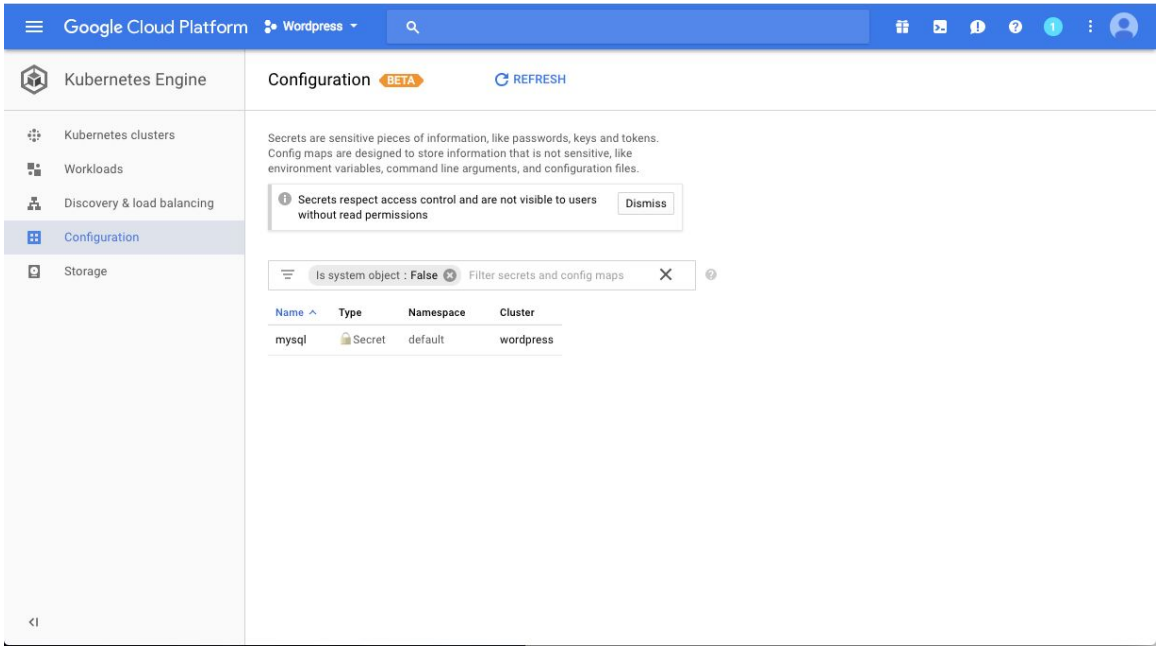
Creating the cluster with get credentials code

```
malyajain@wordpress-187204:~$ gcloud container clusters get-credentials malyajain
ERROR: (gcloud.container.clusters.get-credentials) One of [--zone] must be supplied: Please specify zone.
malyajain@wordpress-187204:~$ gcloud container clusters get-credentials malyajain --project wordpress-187204 --zone us-central1-a
Fetching cluster endpoint and auth data.
kubeconfig entry generated for malyajain.
```

Created the DB for mySQL

```
malyajain@wordpress-187204:~$ kubectl create secret generic mysql --from-literal=password=malyajain
secret "mysql" created
```

mySQL DB of type secret is created in the Cluster



Creating the yaml files from the docker hub stored in github
mysql.yaml is shown below

```
secret "mysql" created
malyajain@wordpress-187204:~$ kubectl create -f mysql.yaml
error: the path "mysql.yaml" does not exist
malyajain@wordpress-187204:~$ kubectl create -f https://github.com/GoogleCloudPlatform/kubernetes-engine-samples/blob/master/wordpress-persistent-disks/mysql.yaml
error: error converting YAML to JSON: yaml: line 312: mapping values are not allowed in this context
malyajain@wordpress-187204:~$ kubectl create -f https://raw.githubusercontent.com/GoogleCloudPlatform/kubernetes-engine-samples/master/wordpress-persistent-disks/mysql.y
aml
deployment "mysql" created
malyajain@wordpress-187204:~$ kubectl create -f https://raw.githubusercontent.com/GoogleCloudPlatform/kubernetes-engine-samples/master/wordpress-persistent-disks/mysql-s
ervice.yaml
service "mysql" created
```

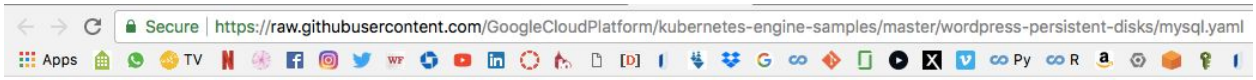
wordpress.yaml and wordpress-service.yaml are shown below

```
malyajain@wordpress-187204:~$ kubectl create -f https://raw.githubusercontent.com/GoogleCloudPlatform/kubernetes-engine-samples/master/wordpress-persistent-disks/wordpre
ss.yaml
deployment "wordpress" created
malyajain@wordpress-187204:~$ kubectl create -f https://raw.githubusercontent.com/GoogleCloudPlatform/kubernetes-engine-samples/master/wordpress-persistent-disks/wordpre
ss-service.yaml
service "wordpress" created
malyajain@wordpress-187204:~$ kubectl get pod -l app=mysql
NAME                     READY     STATUS              RESTARTS   AGE
mysql-3368603707-8v10m   0/1       ContainerCreating   0          48s
malyajain@wordpress-187204:~$ kubectl get service mysql
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
mysql     ClusterIP   10.47.254.149   <none>        3306/TCP   43s
malyajain@wordpress-187204:~$ kubectl get pod -l app=wordpress
NAME                         READY     STATUS    RESTARTS   AGE
wordpress-3479901767-rd8rn   0/1       Pending   0          44s
```

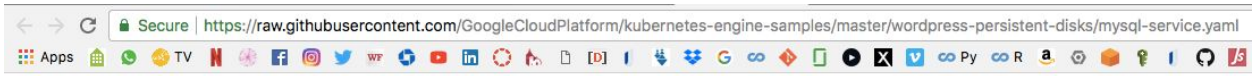
Checking the status of the persistent volume Kubernetes

```
malyajain@wordpress-187204:~$ kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS    CLAIM                   STORAGECLASS   REASON    AGE
task-pv-volume   10Gi       RWO            Retain           Bound     default/task-pv-claim   manual                   30m
```

Showing the .yaml files for the following that were pulled in the container

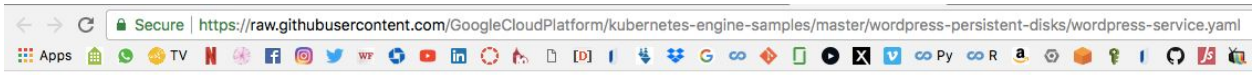



```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql
                  key: password
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-persistent-storage
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-persistent-storage
          gcePersistentDisk:
            pdName: mysql-disk
            fsType: ext4
```




```
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    app: mysql
```


```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
        - image: wordpress
          name: wordpress
          env:
          - name: WORDPRESS_DB_HOST
            value: mysql:3306
          - name: WORDPRESS_DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysql
                key: password
          ports:
            - containerPort: 80
              name: wordpress
          volumeMounts:
            - name: wordpress-persistent-storage
              mountPath: /var/www/html
      volumes:
        - name: wordpress-persistent-storage
          gcePersistentDisk:
            pdName: wordpress-disk
            fsType: ext4
```

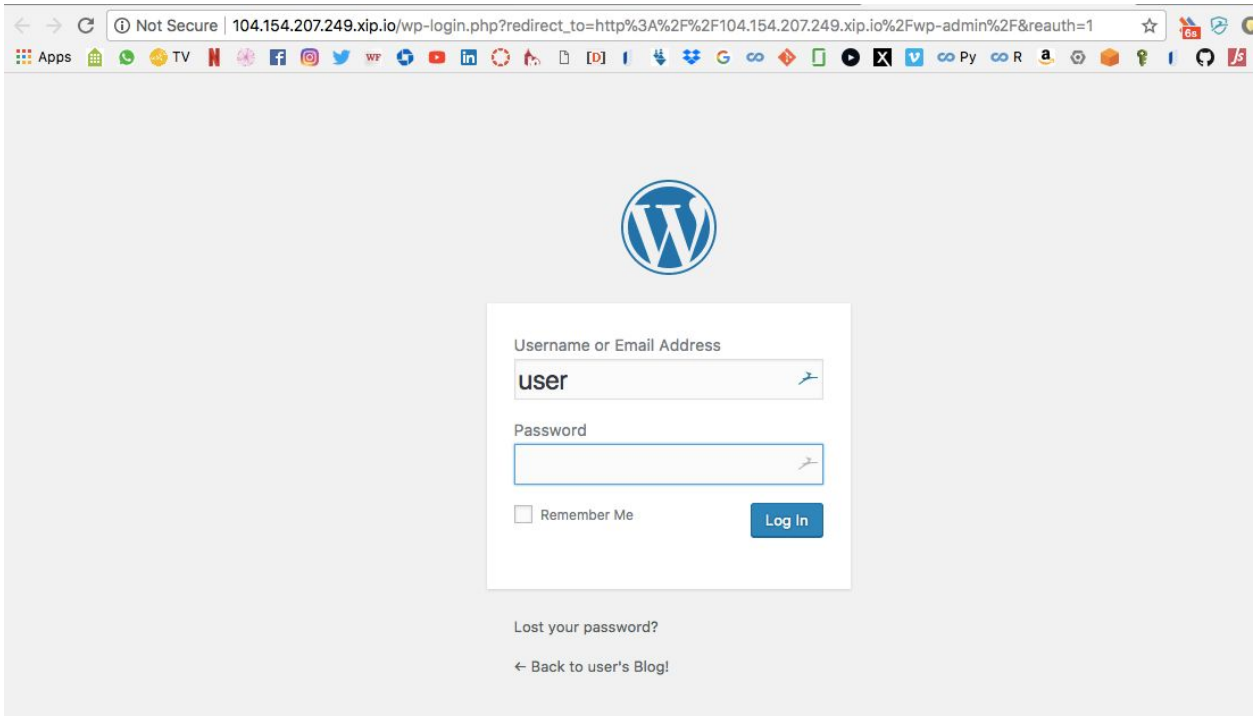


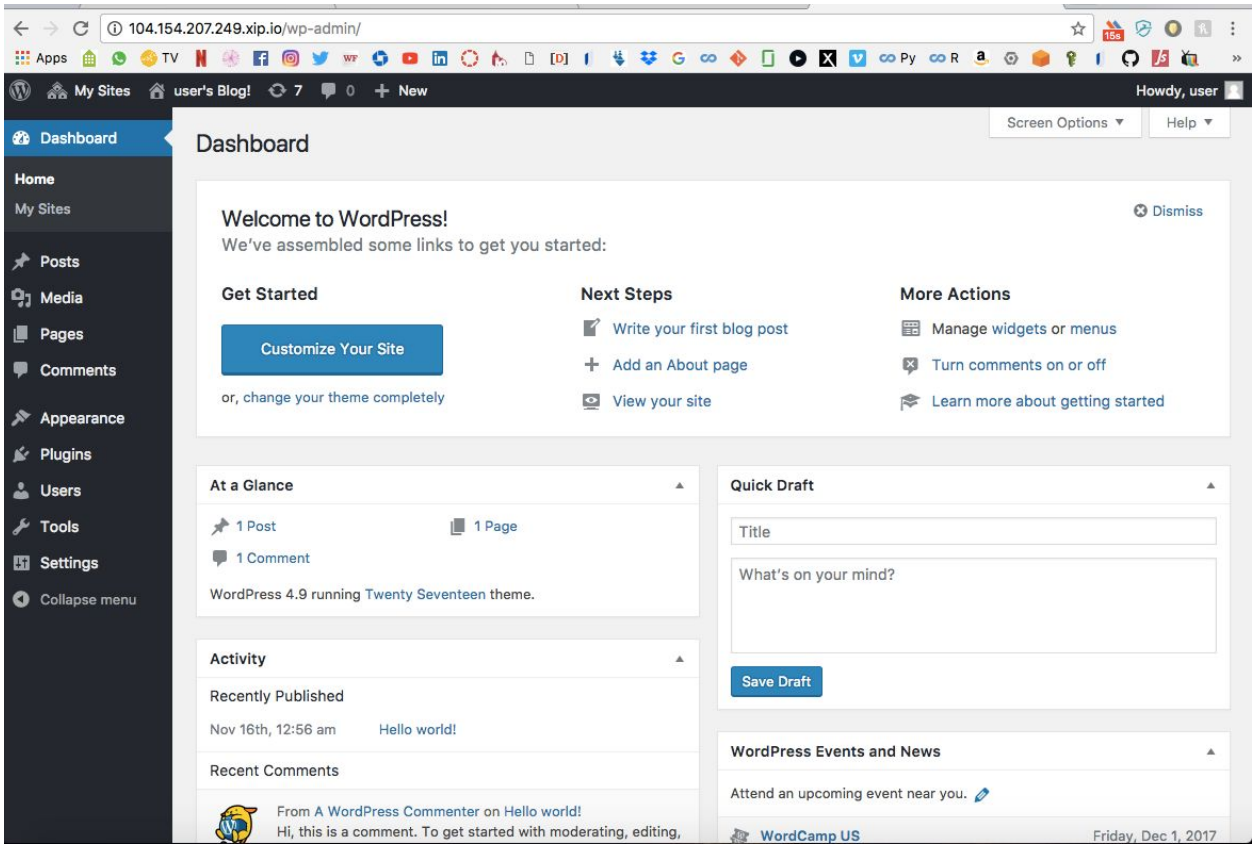

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: wordpress
  name: wordpress
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: wordpress
```

The URL of the website created on Kubernetes - wordpress

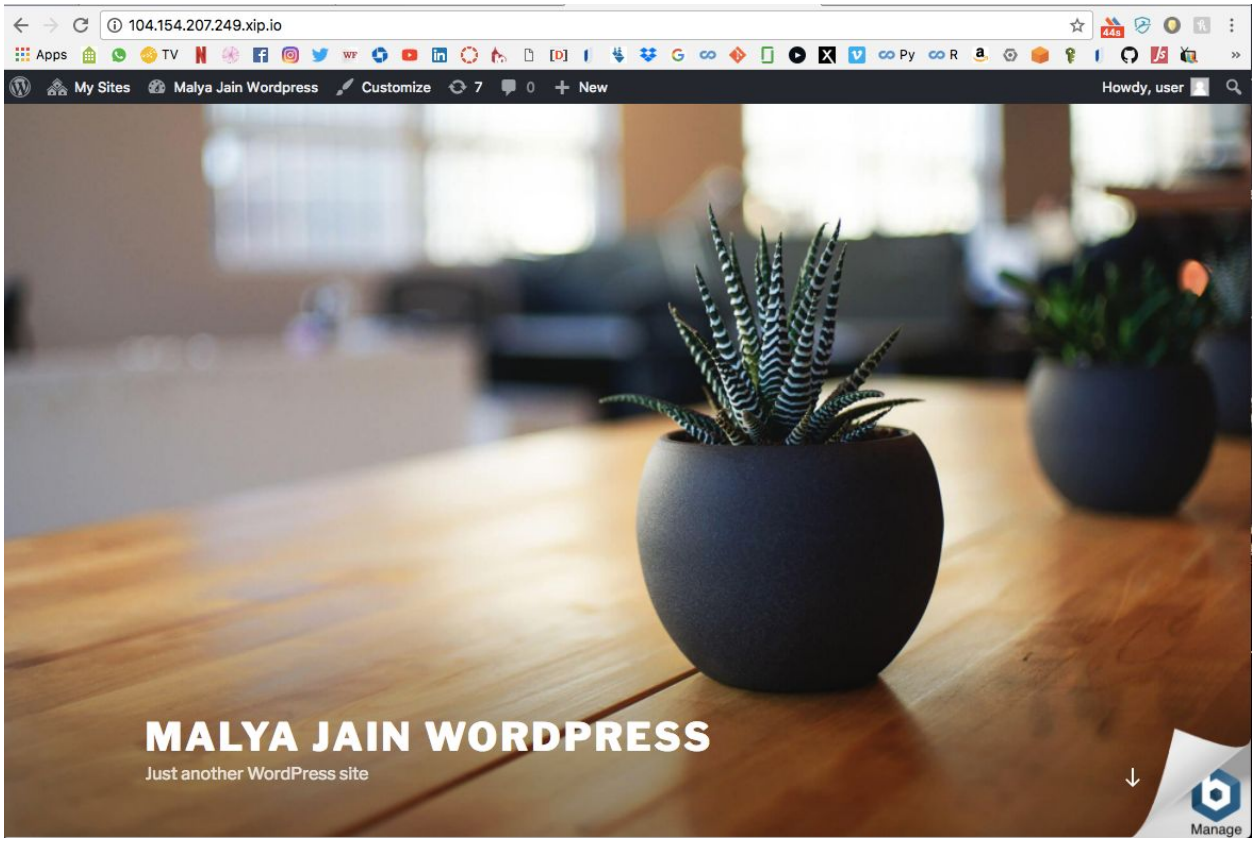
Site address	http://104.154.207.249/
Admin URL	http://104.154.207.249/wp-admin/
Admin user	user
Admin password (Temporary)	Q86VZQ7n2vjd
Instance	wordpress-vm
Instance zone	us-central1-f
Instance machine type	f1-micro

Opening the install.php file gives the login





Wordpress installed on Kubernetes



Docker is better because of the following reasons:
1. It can be easily customized for test and development environments 2. The configurations and dependencies are not maintained internally and not externally 3. As seen above, a docker can be created as a stateless repository. It can then be used to deploy applications free of platforms. The same docker image can be pushed to Amazon Web Services or Microsoft Azure or Google Cloud Platform with simple steps 4. It can easily be linked to GitHub as I have linked my repositories above. If one wants to edit the .yaml files, they can be easily updated with controlled version history enabled 5. Isolation of different resources is possible in docker. The repositories are kept independent of each other 6. The images placed in the docker have a tight security control due to logical segregation between the repositories

I would choose AWS as the docker image container because of the following reasons:
1. Ease of customization and use

2. Better elasticity and scalability is provided as compared to GCP
3. It is available for multiple environments such as Ubuntu or any other Linux VM
4. Multiple services can be used such as a docker can be deployed into EC2, ECS, EBS etc much easily than installation on GCP Kubernetes or GCE
5. No need to create the .yaml files separately in AWS as need to be done and defined in GCP

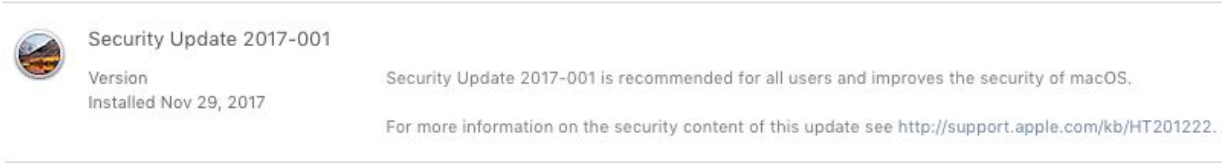
BONUS: OPEN ENDED (YOU DECIDE)

- Pick a Cloud project, define the problem you are trying to solve
- Sample projects:
  - Natural Language API for Sentiment Analysis of movies
  - Spam/Non-Spam email prediction using Classification
  - Alexa for automated speech recognition
  - Using Big Data and Data visualization tools for analysis of a large dataset
- You can implement using Lambda, ML APIs or something new in either of the Cloud Vendors.
  - Document the problem/challenge you are trying to solve and why and how the Cloud helps you
  - You must choose 3 existing components like EC2, S3, RDS, Beanstalk or equivalent of them in Azure or GCP Plus one or more new components like Lambda or ML or Alex etc
  - You can use Big Data from sources like Twitter or other known sources, please provide references.

The Cloud Project I have picked up is **Twitter Sentiment Analysis on AWS**

Background

- Twitter is a microblogging website in which consists of over 330,000,000 active monthly users worldwide (i.e. 330 Million)
- It hosts the voice of millions of users on opinions on anything in the form of 140 character long message called a tweet
- The vast possibilities of Social Media is extremely important for any business these days. Each and every company monitors the social media for how its consumers or customers are responding to the any new feature release or any new product
- Twitter can give real time high accuracy feedback instantly and now is being used as a very powerful tool by every company
- Companies also leverage the vulnerability disclosure on social media by consumers and release the fix.
- For instance, Apple got to know about its major security flaw in logging on as root user without a password. Apple responded to the security flaw and released a zero hour patch for this bug. Reference screenshots as below



Problem I am trying to solve here:

- As I described the importance of the twitter feeds above and how it is being used by the companies these days, but there is a problem which is pretty evident
- As there are 330M active users, the rate at which the consumers express their opinion is really huge and can be classified as big data
- There are thousands of users who tweet every second. Even by searching and analyzing by the hashtags can become very tedious as one can lose track of the tweets. There needs to be a better mechanism to analyze the tweets, see the sentiments of the users if it is inclined towards a positive opinion or negative opinion
- A classic example of this was the US Presidential Elections in 2016. As Twitter is a platform where the users vocalize their opinions, it was taken as the most accurate mechanism to see the reactions of millions of users on the election campaigns.
- Numerous analysts and news channels were analyzing and predicting the sentiments of the citizens of USA in response to the presidential campaigns and were presenting the likelihood of winning of the two candidates
- Most importantly the publicity team members of the presidential candidates all over the world were interpreting the reactions and response of the citizens. This allowed the presidential teams to improve the campaigns

Solution:

- Provide an easy mechanism for analyzing the tweets based on a hashtag and proving a real-time analysis of the positive and negative tweets

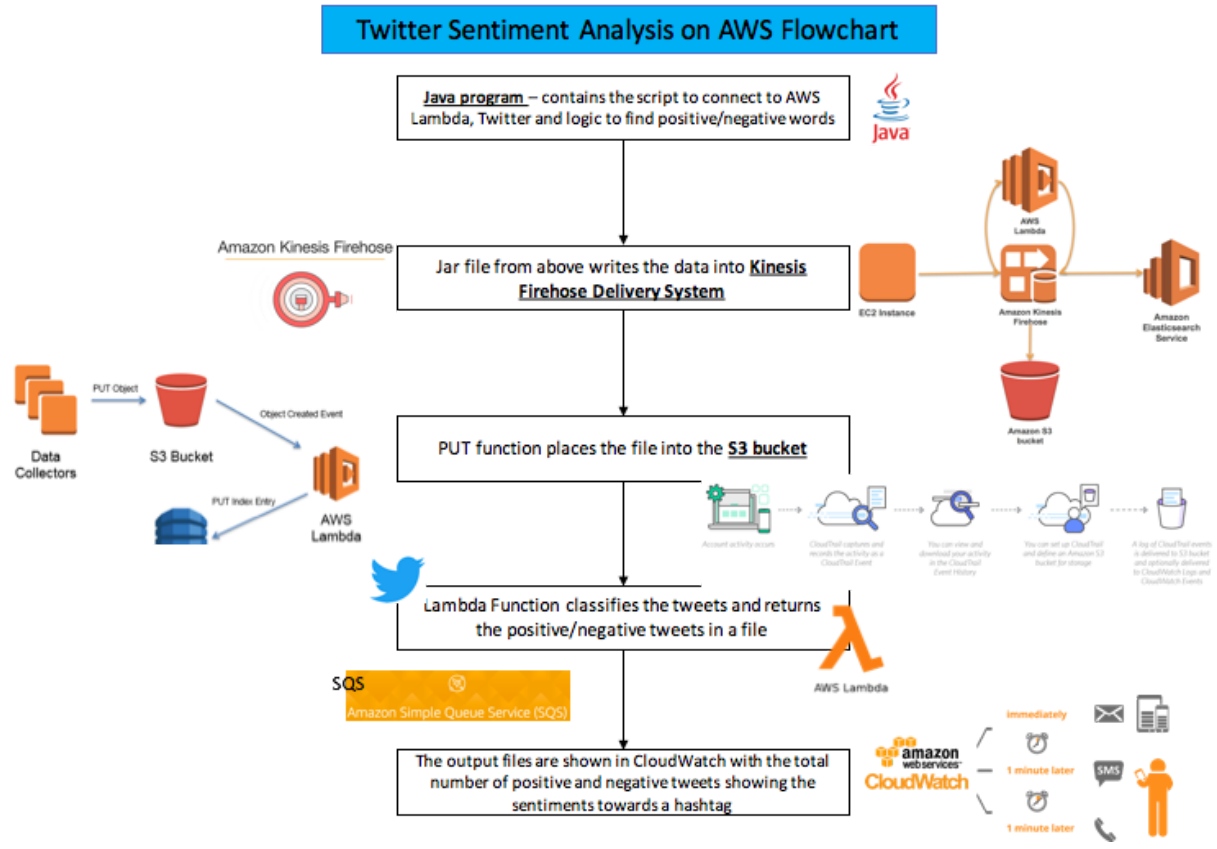
AWS Functionalities used:

1. Lambda Function
2. S3
3. Kinesis Firehose
4. CloudWatch
5. SQS (Simple Queue Service)
6. CloudTrail

Workflow:

The java program writes the data to kinesis Firehose delivery stream -> this writes a file to S3 bucket -> this triggers a lambda function to classify the file as positive/negative file -> watch the output in cloudwatch.

Flowchart:



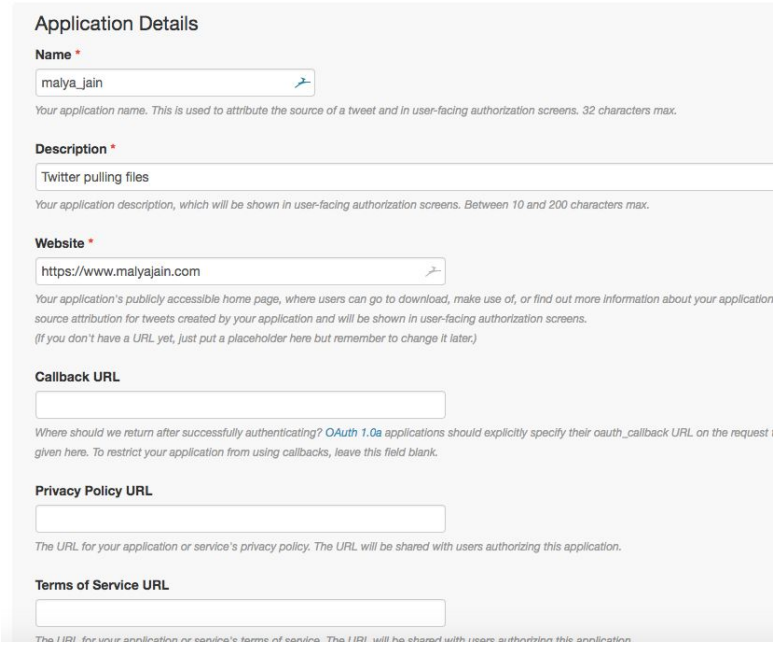
Important URLs:

Lambda ARN:

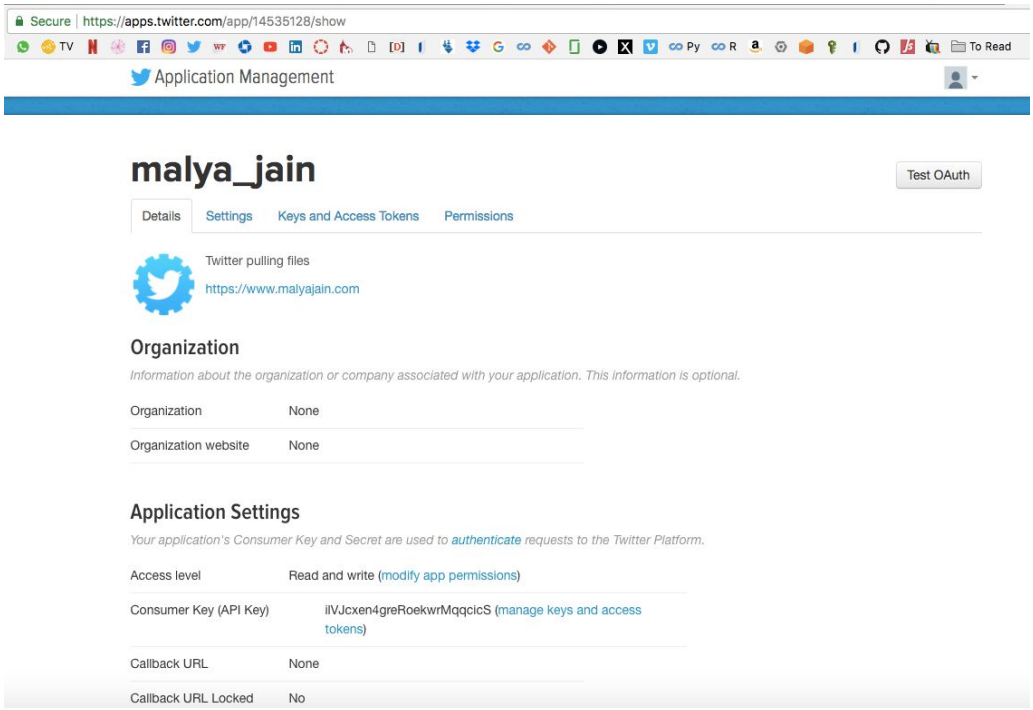
arn:aws:lambda:us-west-2:012539484088:function:test-s3-lambda-invoke

S3 Bucket with the file written:
https://s3-us-west-2.amazonaws.com/test-s3-lambda-invoke/2017/12/02/17/twitter-data-to-s3-1-2017-12-02-17-35-02-c80d119b-15ab-44a2-89c3-edf1f86ab2cb
CloudTrail:
https://us-west-2.console.aws.amazon.com/cloudtrail/home?region=us-west-2#/configuration/arn:aws:cloudtrail:us-west-2:012539484088:trail@test-s3-lambda-invoke-trail
Kinesis Firehose:
https://us-west-2.console.aws.amazon.com/firehose/home?region=us-west-2#/details/twitter-data-to-s3
SQS:
https://sqs.us-west-2.amazonaws.com/012539484088/test-sns-lambda-publish-dlq
https://sqs.us-west-2.amazonaws.com/012539484088/test-s3-lambda-invoke-dlq

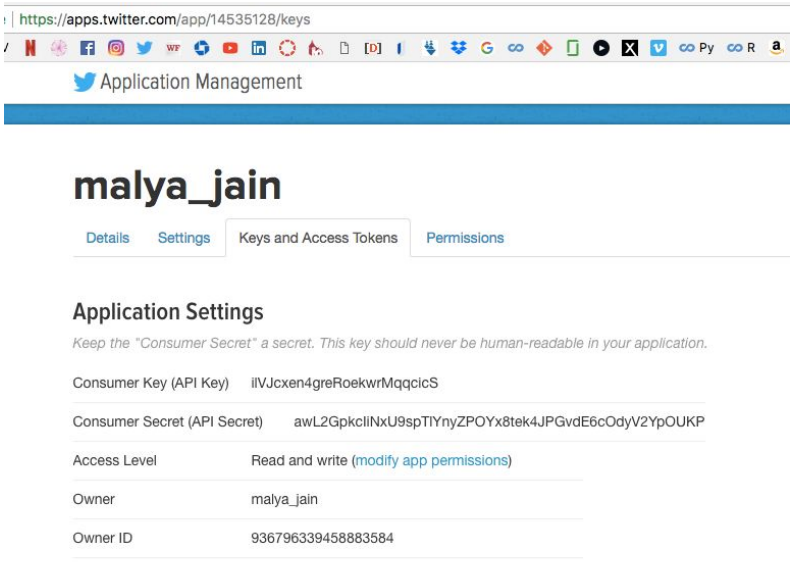
Created the Twitter application under the personal twitter account. Created a sample app called malya_jain used for twitter putting files. Put the website as www.malyajain.com



Once I create the application, it will directly give me 4 codes namely: Access Level, Consumer Key

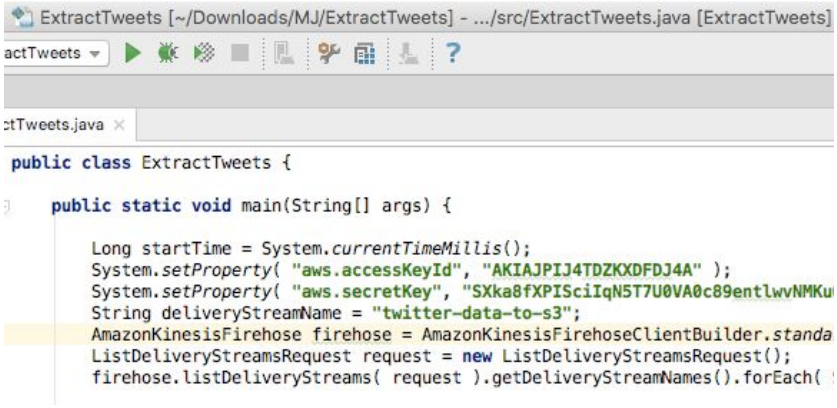


The other two keys are shown as Consumer Key and Consumer Secret key as used in this screenshot below



App-only authentication	https://api.twitter.com/oauth2/token
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token

Now the next step is to create a .jar file with the code.
The configuration is created into the jar file. The snippet below shows how my code is connected to the AWS account. Below is the AWS Access Key ID and AWS Secret Key.



This snippet below is creating a new configuration builder and setting the country name as United States for pulling the twitter feeds. The snippet now below shows how I connected this code to my twitter account.

```
ConfigurationBuilder cb = new ConfigurationBuilder();
cb.setDebugEnabled( true )
  .setOAuthConsumerKey( "ilVJcxen4greRoekwrMqqcicS" )
  .setOAuthConsumerSecret( "awL2GpkcIiNxU9spTlYnyZPOYx8tek4JPGvdE6cOdyV2YpOUKP" )
  .setOAuthAccessToken( "936796339458883584-50DCDR4mk4fqlHiu2fHbptwVNQ85AR7" )
  .setOAuthAccessTokenSecret( "7y7TIvZ542FrTjrdMdDn8q4F1sF8xdYoq9KGkh7KLpe8N" );
Twitter twitter = new TwitterFactory(cb.build()).getInstance();
int woeid = 1;
String countryName = "United States";
```

Next, this function gets the list of most happening hashtags on twitter at the particular time.
It is now generating a random number for selecting the most popular hashtag out of the group of hashtags in on Twitter.
As the country above has been set to United States of America, I have enabled to get the tweets from one particular city in order to extract the files only for that city for the most happening hashtag.
This functionality has been enabled so that only english language tweets are obtained in the output. There was a problem faced previously, when the code was returning only the most

happening hashtags, this was not required because our list of positive, negative and stop words only contains english words.

```
String searchParameter = "";
try {
    Trends currentTrends = twitter.getPlaceTrends( woeid );
    int randomNumber = ThreadLocalRandom.current().nextInt( origin: 0, cur
    System.out.println( randomNumber + " " + currentTrends.getTrends()[r
    searchParameter = currentTrends.getTrends()[randomNumber].getName();
} catch ( TwitterException e ) {
    e.printStackTrace();
```

This snippet prints the output as the getting the text of the tweet. This puts the record into Kinesis Firehose delivery stream. The stream is a buffer that buffers for time (upto 240 seconds) or for capacity (upto 128 MB). Once it reaches either capacity, it writes that file to S3 bucket and starts the buffering process once again.

```
System.out.println( "Querying for trend: " +  searchParameter);
Query query = new Query( searchParameter );
QueryResult queryResult;
try {
    do {
        queryResult = twitter.search( query );
        final List<Status> tweets = queryResult.getTweets();
        tweets.forEach( tweet -> {
            System.out.println(tweet.getText() );
            String tweetWithEOL = tweet.getText() + "\n";
            PutRecordRequest putRecordRequest = new PutRecordRequest();
            putRecordRequest.setDeliveryStreamName( deliveryStreamName );
            Record record = new Record();
            record.setData( ByteBuffer.wrap( tweetWithEOL.getBytes() ) );
            putRecordRequest.setRecord( record );
            firehose.putRecord( putRecordRequest );
        } );
        query = queryResult.nextQuery();
    } while ( query != null );
```

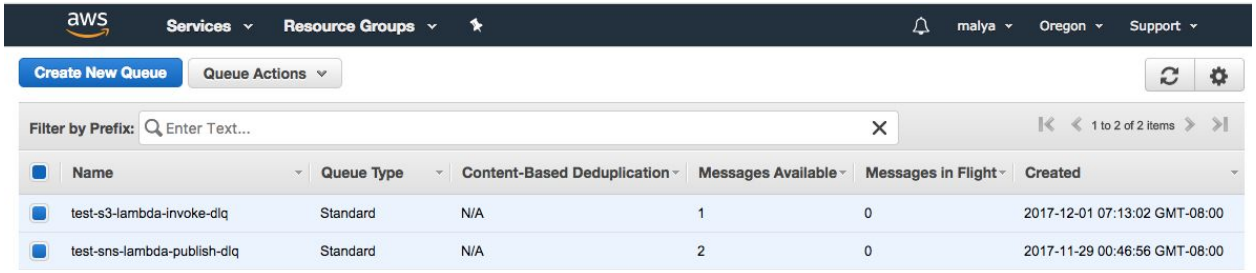
The output is shown as below. The extracted tweets file queried for the trend #TaxScamBill; which is the most happening hashtag at the point of running this file. Further, from the random number generator, the selected city is Dallas-Ft. Worth.

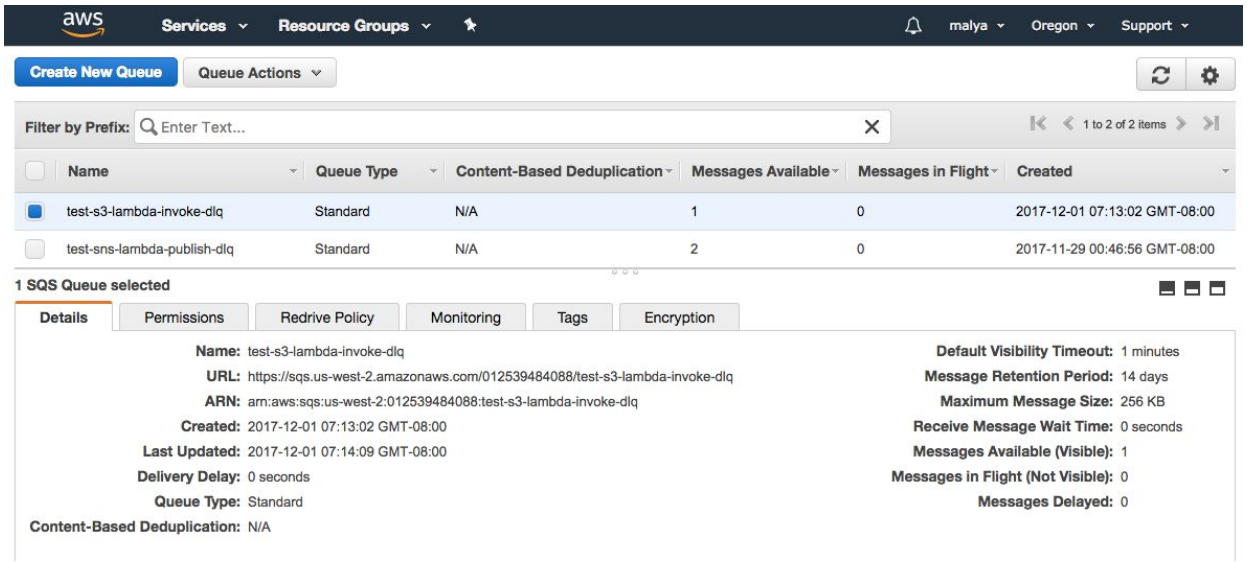
```
ExtractTweets
/Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java ...
objc[47645]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (0x101a9d4c0)
twitter-data-to-s3
woeid is: 2388929 name is: Dallas-Ft. Worth
6 #TaxScamBill
Querying for trend: #TaxScamBill
RT @ProudResister: WHILE YOU WERE SLEEPING, the GOP passed their #TaxScamBill, which will rip healthcare from 13 million, increase the debt…
I'd really love to know from @SenateMajLdr and @SenateGOP what oil drilling in Alaska has to do with supposed tax r… https://t.co/guvTKoiAGB
RT @ProfSybill: ⛽The #TaxScamBill #GOPTaxBill is NOT law yet. ⛽
```

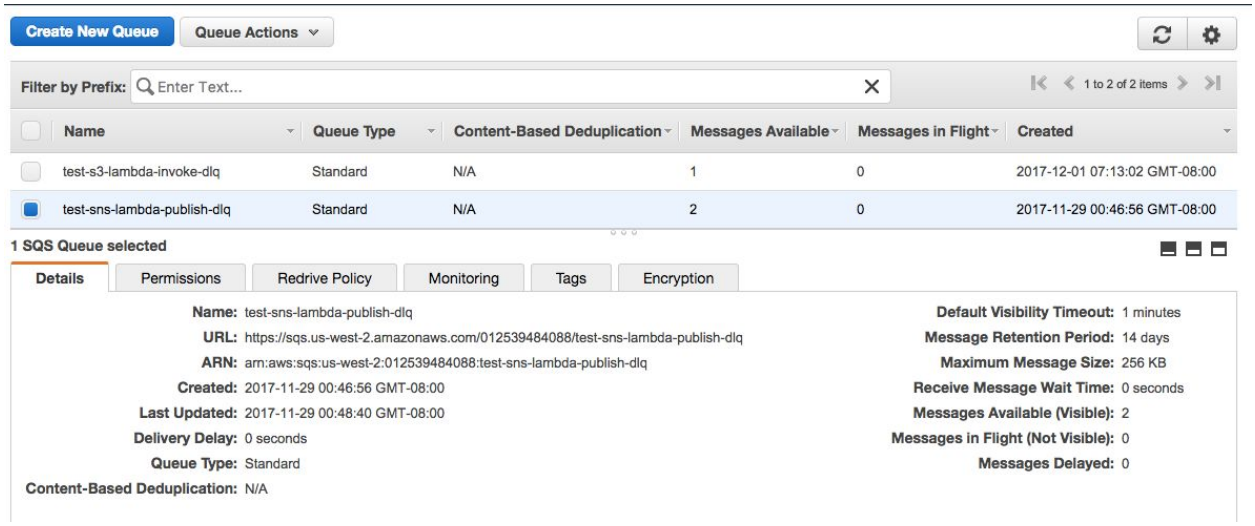
Scrolling down to the end of the output file shows that the code ran for a total of 30269 milliseconds

```
#TaxScamBill When Liberal dreams come true...until later that evening and ABC retracts the story #FakeNews

https://t.co/9qKAsYmhd0
RT @Alyssa_Milano: You think this #TaxScamBill is bad? Wait until they take the internet from you and give
RT @EdKrassen: Within 10 years, the Trump-Republican #TaxScamBill will cost the middle class $5.3 billion &
RT @ProudResister: The GOP just gave a giant MIDDLE FINGER to the Middle Class and passed their #TaxScamBil

13 million will lose healthc…
RT @susi_debra: Congratulations, Middle Class. Here's your cut of the new #TaxScamBill https://t.co/CrkljKs
RT @AmandiOnAir: Feeling powerless about the @GOP #TaxScamBill?

Grab a pen & write this date down: NOVEMBER 6, 2018

It's a more importa…
RT @Pink_About_it: #TaxScamBill

Abc news:

Russia hacked the Senate votes to give Americans a tax cut

Also ABC news 7 hours later:

We…
Total time is: 30269
```

Now we use the SQS (Simple Queue Service) in this where the notification is being sent to the AWS instance.

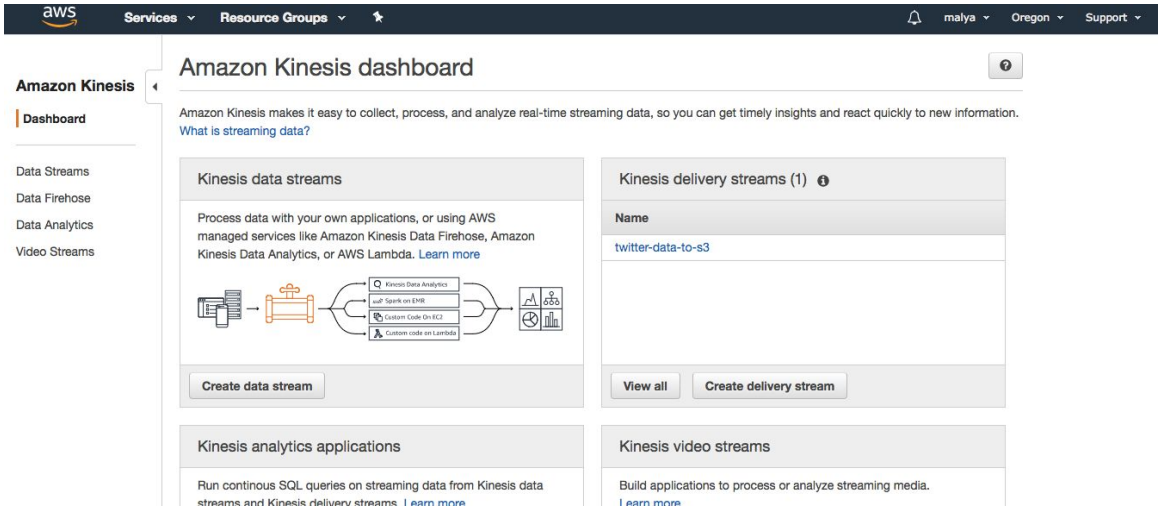




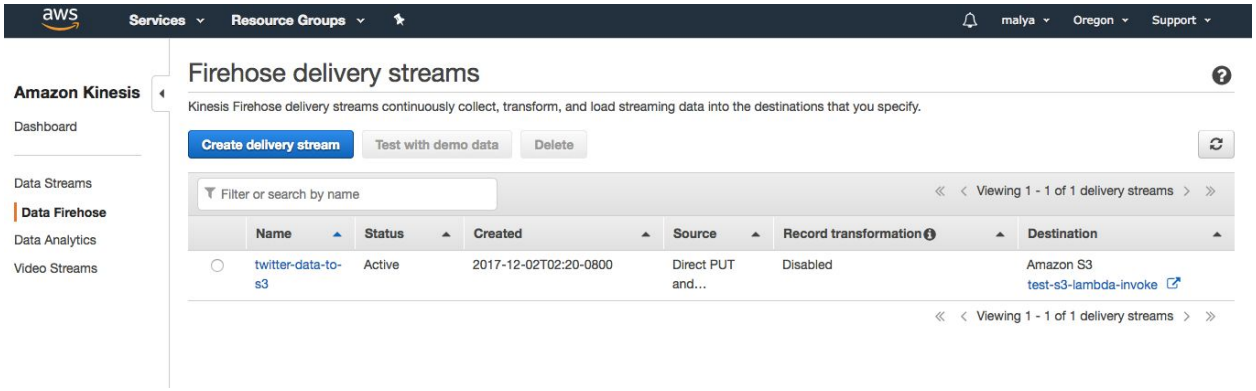


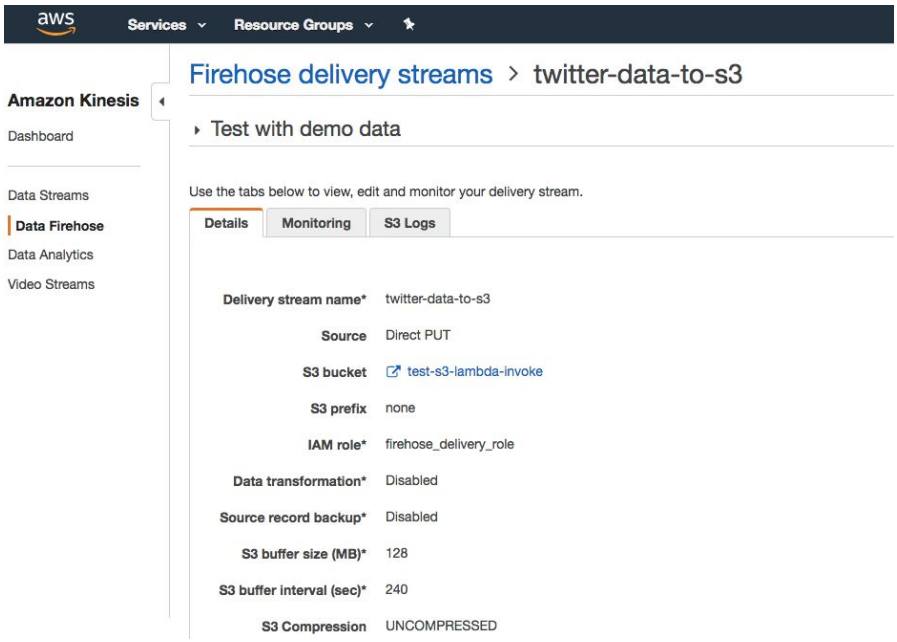
Another functionality used in the project is Kinesis Firehose. This step comes before the file is written into the S3 bucket. This acts as a buffer zone before sending the files onto the S3 bucket.

Buffer is stored upto 240 seconds or until the file becomes 128 mb in size.

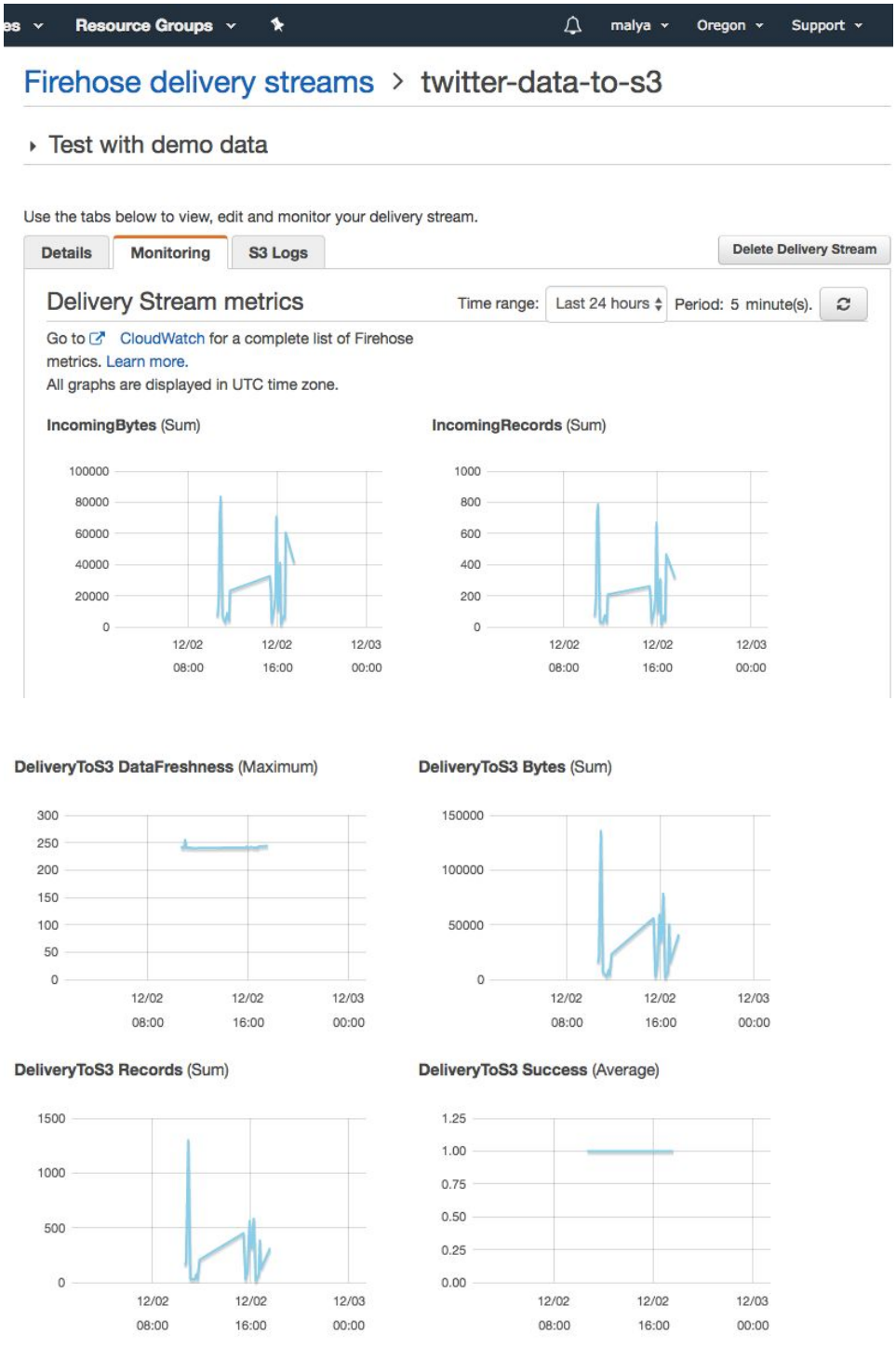


A delivery system called twitter-data-to-s3 is created in Kinesis Data Firehose

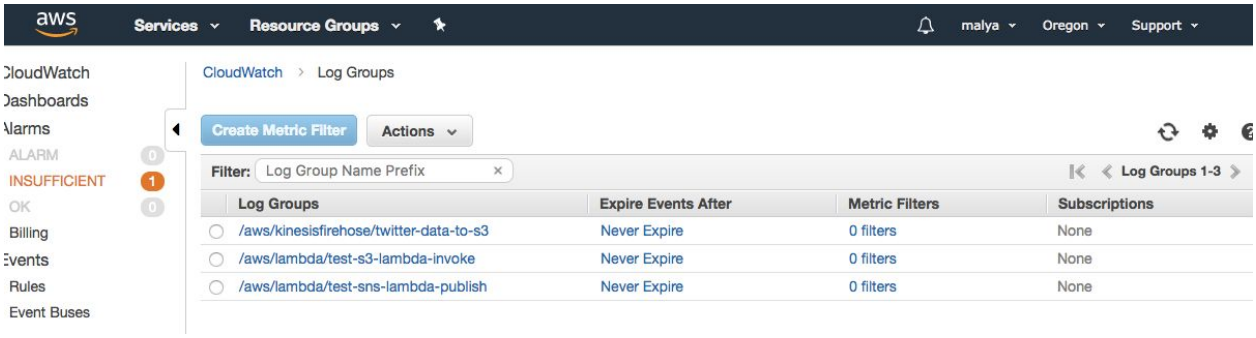




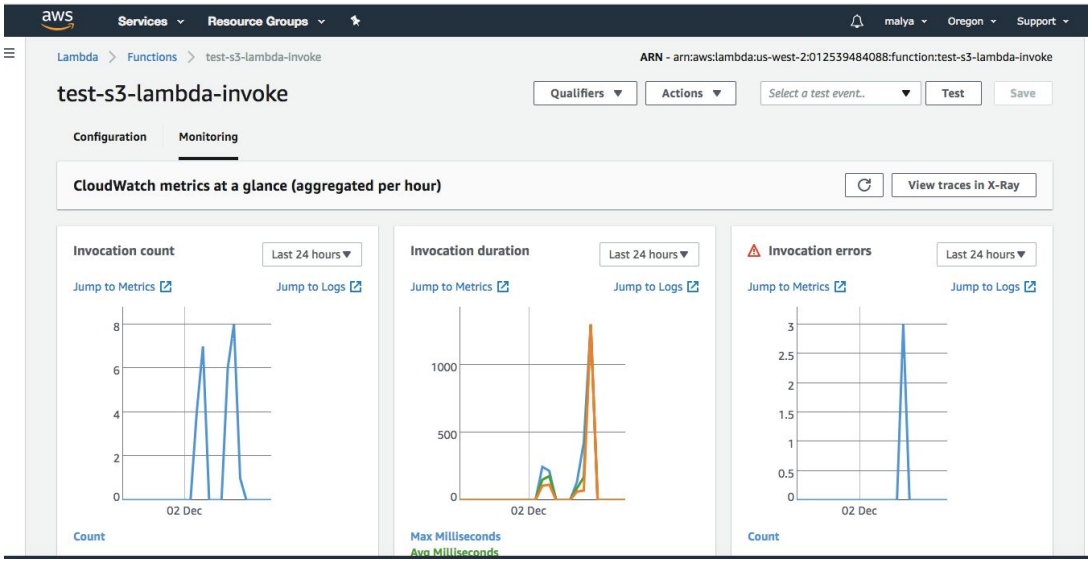
This enables us to even monitor and logs being sent to the S3 bucket.
We can even go to the monitoring tab and see how the Delivery Stream Metrics and see that at 16:00 (UTC) the files were to send to the S3 bucket



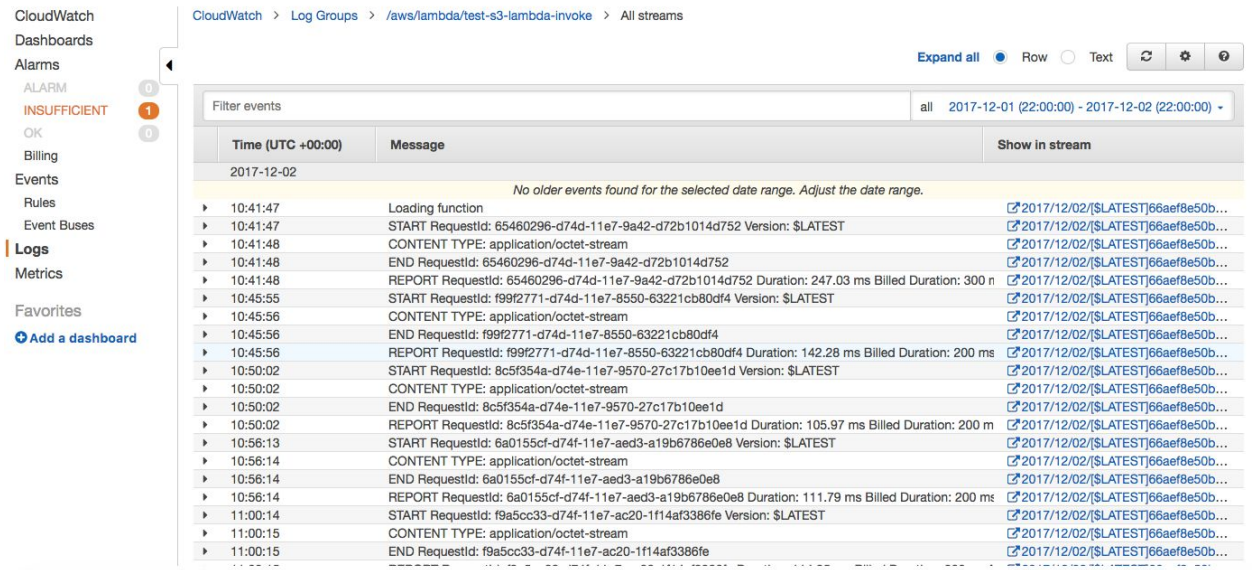
Another functionality used is the Lambda, that does the processing when a new file is uploaded/written into the s3 bucket. The lambda processing logs is what cloudwatch shows (see for log group test-s3-lambda-invoke).



Selecting the test-s3-lambda-invoke functionality provides us with the cloudwatch metrics. This function is the core of classifying tweets as either positive or negative.



Looking at all the events, gives us the following logs. As can be seen from the screenshot, the function was run on 12/02/17 at 10:41 (This time is in UTC)



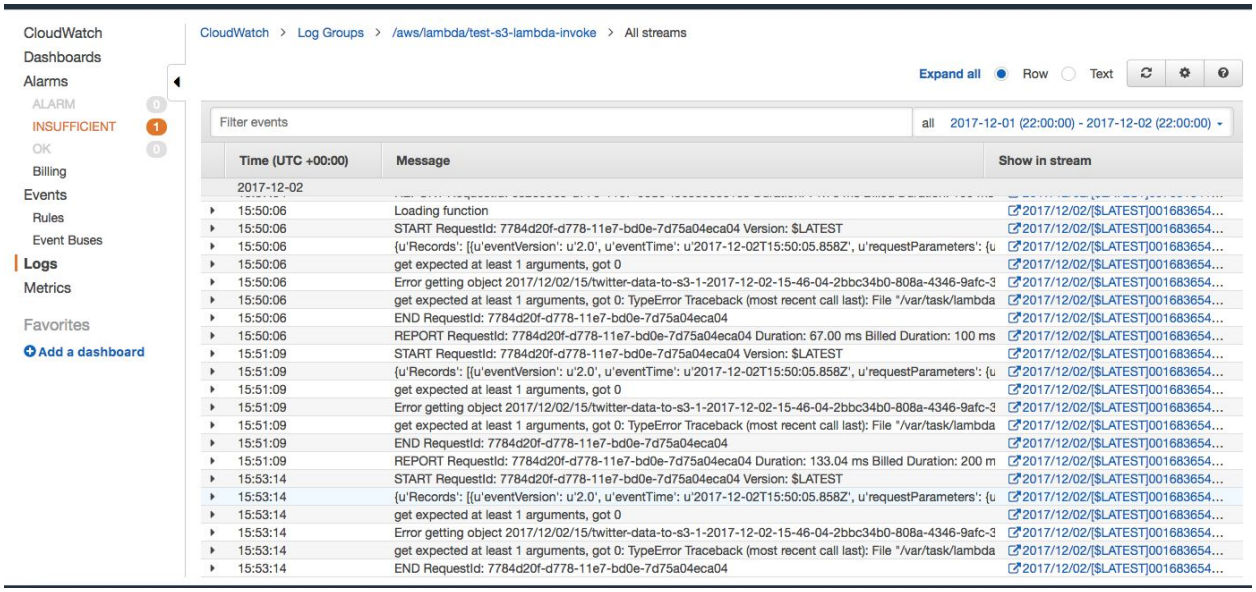
CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 1
OK 0
Billing
Events
Rules
Event Buses
Logs
Metrics
Favorites
Add a dashboard

CloudWatch > Log Groups > /aws/lambda/test-s3-lambda-invoke > All streams

Expand all | Row | Text

Filter events | all | 2017-12-01 (22:00:00) - 2017-12-02 (22:00:00)

Time (UTC +00:00)	Message	Show in stream
2017-12-02		
	No older events found for the selected date range. Adjust the date range.	
10:41:47	Loading function	2017/12/02/[$LATEST]66aef8e50b...
10:41:47	START RequestId: 65460296-d74d-11e7-9a42-d72b1014d752 Version: $LATEST	2017/12/02/[$LATEST]66aef8e50b...
10:41:48	CONTENT TYPE: application/octet-stream	2017/12/02/[$LATEST]66aef8e50b...
10:41:48	END RequestId: 65460296-d74d-11e7-9a42-d72b1014d752	2017/12/02/[$LATEST]66aef8e50b...
10:41:48	REPORT RequestId: 65460296-d74d-11e7-9a42-d72b1014d752 Duration: 247.03 ms Billed Duration: 300 m	2017/12/02/[$LATEST]66aef8e50b...
10:45:55	START RequestId: f99f2771-d74d-11e7-8550-63221cb80df4 Version: $LATEST	2017/12/02/[$LATEST]66aef8e50b...
10:45:56	CONTENT TYPE: application/octet-stream	2017/12/02/[$LATEST]66aef8e50b...
10:45:56	END RequestId: f99f2771-d74d-11e7-8550-63221cb80df4	2017/12/02/[$LATEST]66aef8e50b...
10:45:56	REPORT RequestId: f99f2771-d74d-11e7-8550-63221cb80df4 Duration: 142.28 ms Billed Duration: 200 ms	2017/12/02/[$LATEST]66aef8e50b...
10:50:02	START RequestId: 8c5f354a-d74e-11e7-9570-27c17b10ee1d Version: $LATEST	2017/12/02/[$LATEST]66aef8e50b...
10:50:02	CONTENT TYPE: application/octet-stream	2017/12/02/[$LATEST]66aef8e50b...
10:50:02	END RequestId: 8c5f354a-d74e-11e7-9570-27c17b10ee1d	2017/12/02/[$LATEST]66aef8e50b...
10:50:02	REPORT RequestId: 8c5f354a-d74e-11e7-9570-27c17b10ee1d Duration: 105.97 ms Billed Duration: 200 m	2017/12/02/[$LATEST]66aef8e50b...
10:56:13	START RequestId: 6a0155cf-d74f-11e7-aed3-a19b6786e0e8 Version: $LATEST	2017/12/02/[$LATEST]66aef8e50b...
10:56:14	CONTENT TYPE: application/octet-stream	2017/12/02/[$LATEST]66aef8e50b...
10:56:14	END RequestId: 6a0155cf-d74f-11e7-aed3-a19b6786e0e8	2017/12/02/[$LATEST]66aef8e50b...
10:56:14	REPORT RequestId: 6a0155cf-d74f-11e7-aed3-a19b6786e0e8 Duration: 111.79 ms Billed Duration: 200 ms	2017/12/02/[$LATEST]66aef8e50b...
11:00:14	START RequestId: f9a5cc33-d74f-11e7-ac20-1f14af3386fe Version: $LATEST	2017/12/02/[$LATEST]66aef8e50b...
11:00:15	CONTENT TYPE: application/octet-stream	2017/12/02/[$LATEST]66aef8e50b...
11:00:15	END RequestId: f9a5cc33-d74f-11e7-ac20-1f14af3386fe	2017/12/02/[$LATEST]66aef8e50b...



CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 1
OK 0
Billing
Events
Rules
Event Buses
Logs
Metrics
Favorites
Add a dashboard

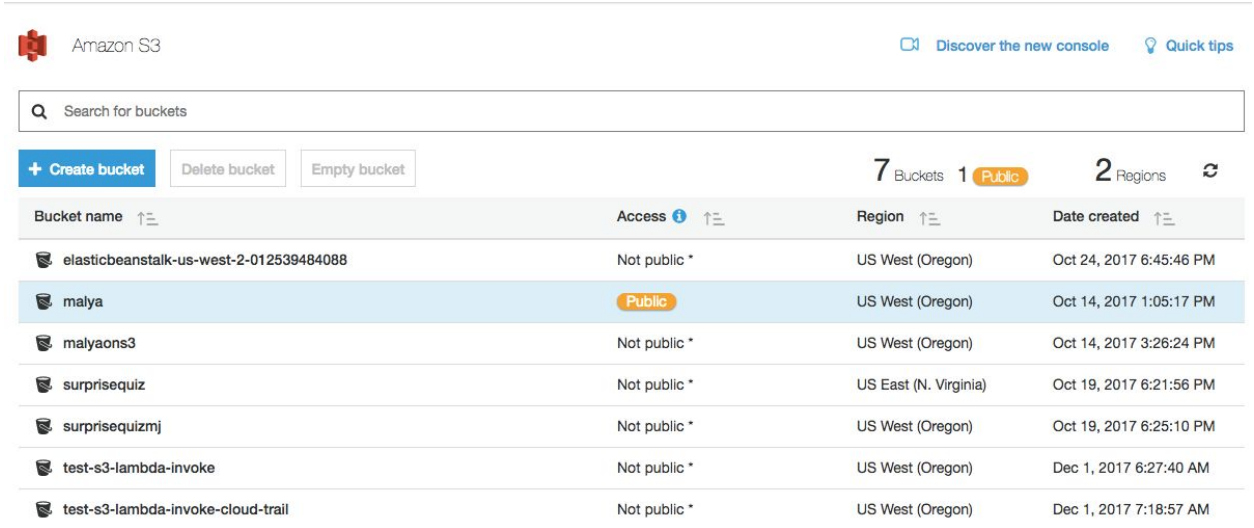
CloudWatch > Log Groups > /aws/lambda/test-s3-lambda-invoke > All streams

Expand all | Row | Text

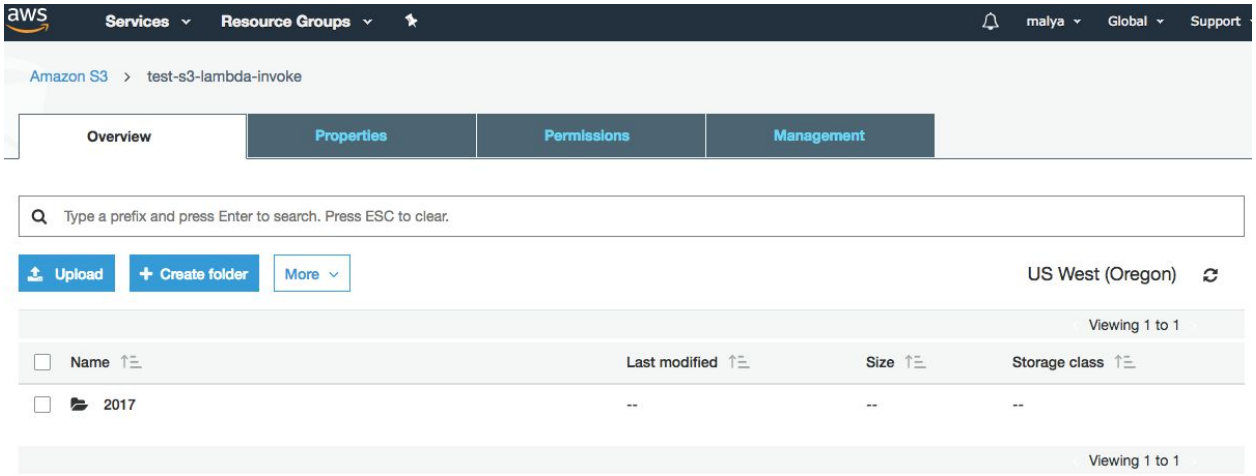
Filter events | all | 2017-12-01 (22:00:00) - 2017-12-02 (22:00:00)

Time (UTC +00:00)	Message	Show in stream
2017-12-02		
15:50:06	Loading function	2017/12/02/[$LATEST]001683654...
15:50:06	START RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04 Version: $LATEST	2017/12/02/[$LATEST]001683654...
15:50:06	{u'Records': [{u'eventVersion': u'2.0', u'eventTime': u'2017-12-02T15:50:05.858Z', u'requestParameters': {u	2017/12/02/[$LATEST]001683654...
15:50:06	get expected at least 1 arguments, got 0	2017/12/02/[$LATEST]001683654...
15:50:06	Error getting object 2017/12/02/15/twitter-data-to-s3-1-2017-12-02-15-46-04-2bbc34b0-808a-4346-9afc-3	2017/12/02/[$LATEST]001683654...
15:50:06	get expected at least 1 arguments, got 0: TypeError Traceback (most recent call last): File "/var/task/lambda	2017/12/02/[$LATEST]001683654...
15:50:06	END RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04	2017/12/02/[$LATEST]001683654...
15:50:06	REPORT RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04 Duration: 67.00 ms Billed Duration: 100 ms	2017/12/02/[$LATEST]001683654...
15:51:09	START RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04 Version: $LATEST	2017/12/02/[$LATEST]001683654...
15:51:09	{u'Records': [{u'eventVersion': u'2.0', u'eventTime': u'2017-12-02T15:50:05.858Z', u'requestParameters': {u	2017/12/02/[$LATEST]001683654...
15:51:09	get expected at least 1 arguments, got 0	2017/12/02/[$LATEST]001683654...
15:51:09	Error getting object 2017/12/02/15/twitter-data-to-s3-1-2017-12-02-15-46-04-2bbc34b0-808a-4346-9afc-3	2017/12/02/[$LATEST]001683654...
15:51:09	get expected at least 1 arguments, got 0: TypeError Traceback (most recent call last): File "/var/task/lambda	2017/12/02/[$LATEST]001683654...
15:51:09	END RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04	2017/12/02/[$LATEST]001683654...
15:51:09	REPORT RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04 Duration: 133.04 ms Billed Duration: 200 m	2017/12/02/[$LATEST]001683654...
15:53:14	START RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04 Version: $LATEST	2017/12/02/[$LATEST]001683654...
15:53:14	{u'Records': [{u'eventVersion': u'2.0', u'eventTime': u'2017-12-02T15:50:05.858Z', u'requestParameters': {u	2017/12/02/[$LATEST]001683654...
15:53:14	get expected at least 1 arguments, got 0	2017/12/02/[$LATEST]001683654...
15:53:14	Error getting object 2017/12/02/15/twitter-data-to-s3-1-2017-12-02-15-46-04-2bbc34b0-808a-4346-9afc-3	2017/12/02/[$LATEST]001683654...
15:53:14	get expected at least 1 arguments, got 0: TypeError Traceback (most recent call last): File "/var/task/lambda	2017/12/02/[$LATEST]001683654...
15:53:14	END RequestId: 7784d20f-d778-11e7-bd0e-7d75a04eca04	2017/12/02/[$LATEST]001683654...

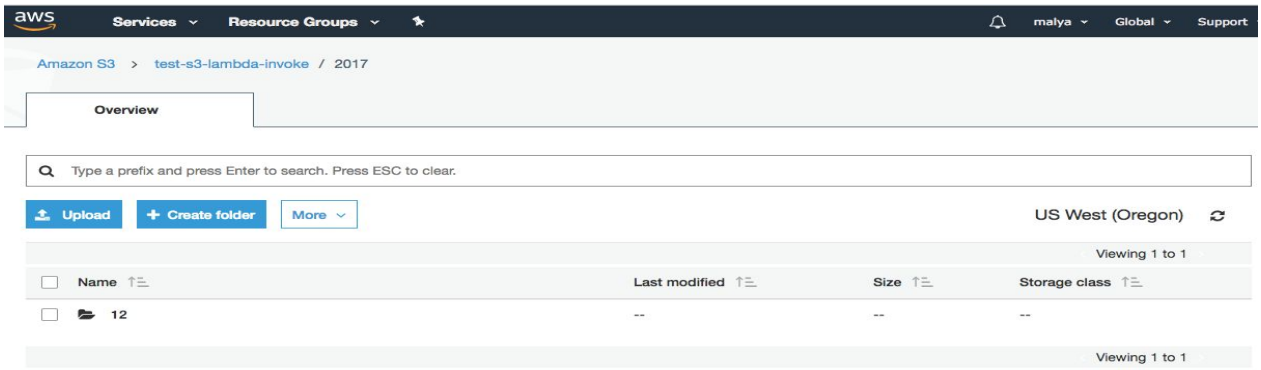
Since the integration of this code is into S3 bucket, the input file is created for lambda and stored in the bucket called test-s3-lambda-invoke
S3 doesn't store the output file, it stores the input file for lambda. For now the output is in cloudwatch logs only.

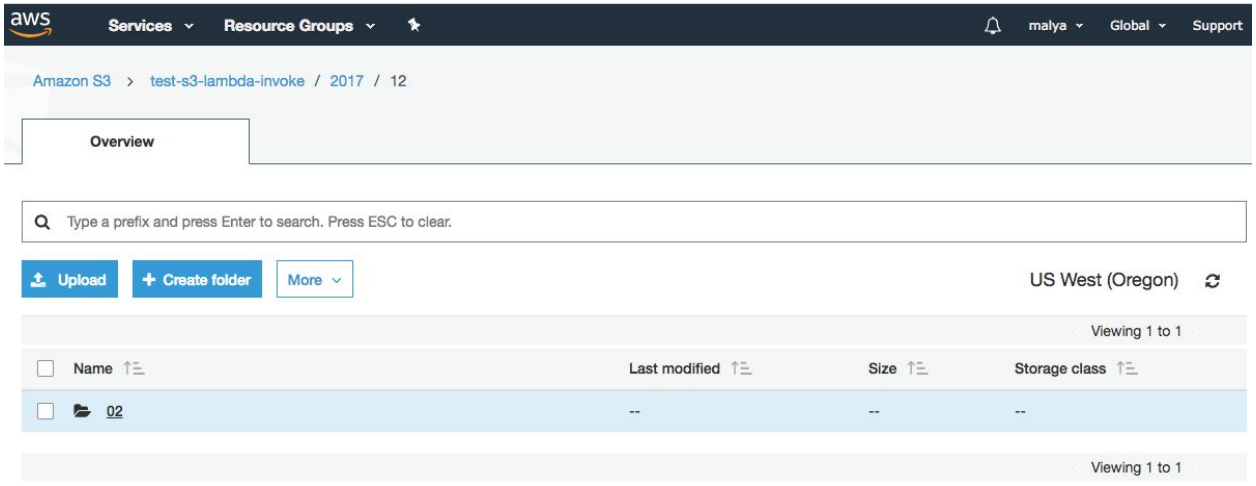


Navigating to the required folder: Year 2017

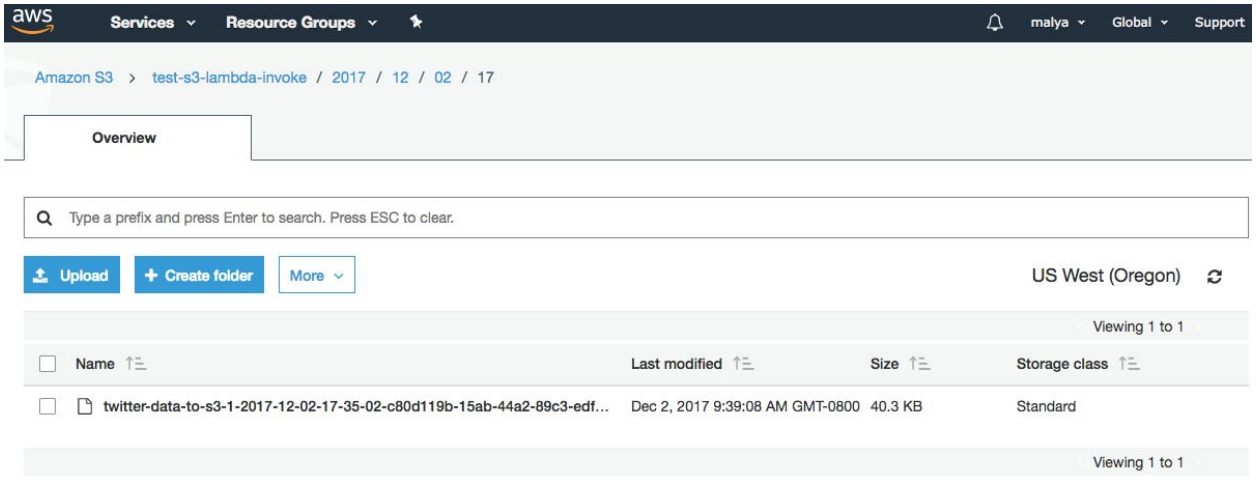


Month 12:

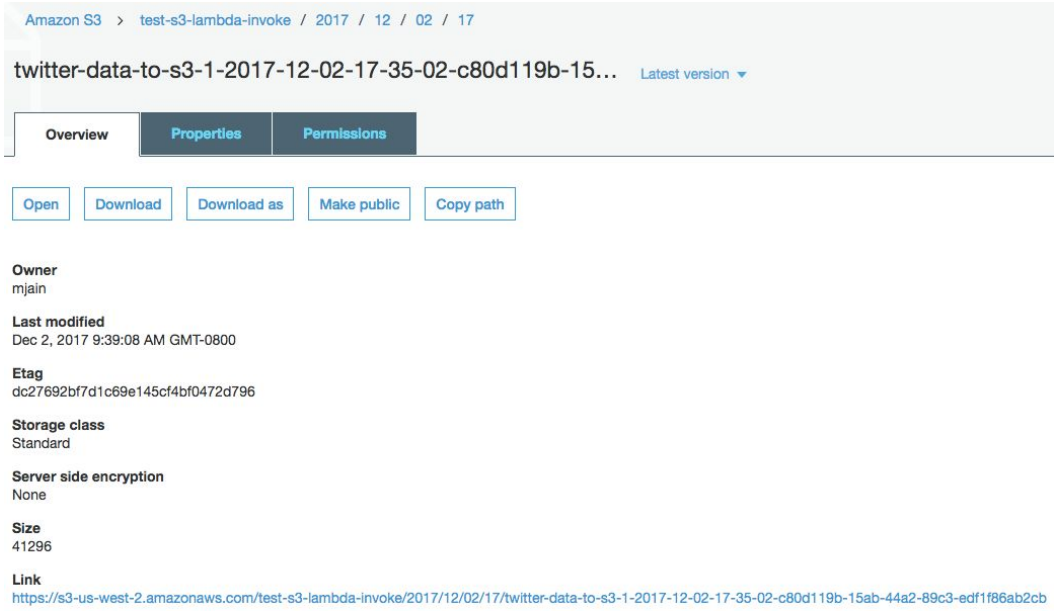




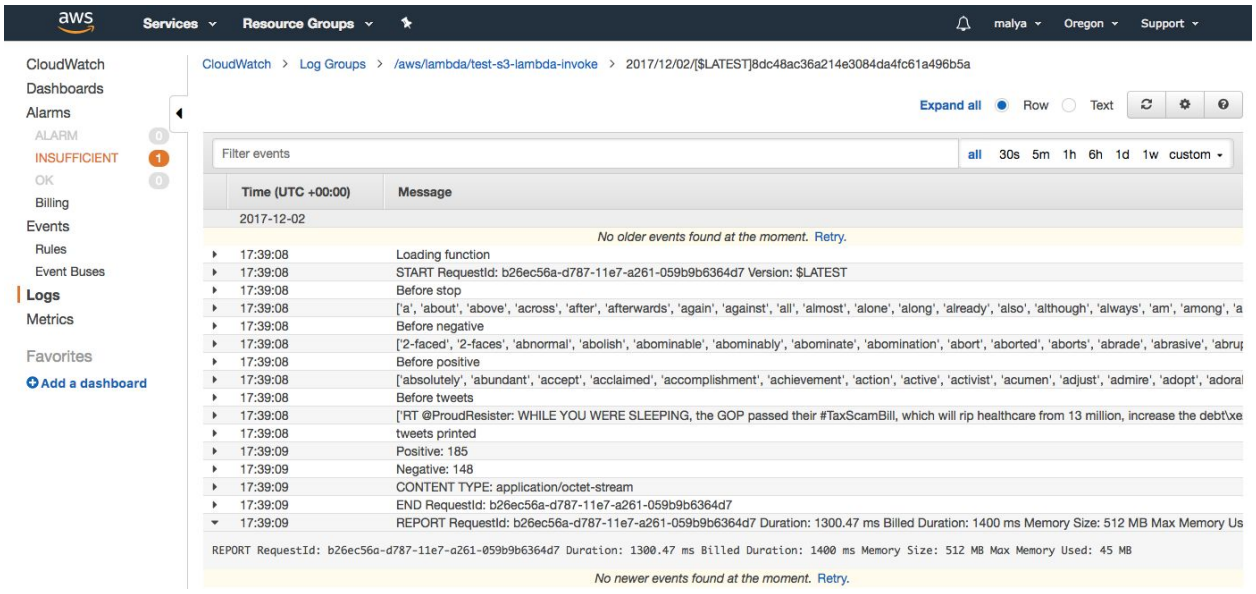
This file was created on December 2, 2017 at 9:39 AM, at the time of running the program file in lambda



More details of the file are as below:



The final output of the file is created as in the CloudWatch as we can see in the screenshot below. For the output file with the hashtag #TaxScamBill analysed for the city of Dallas-Ft Worth in USA, the positive tweets are 185 and negative tweets are 148. This analysis shows that the sentiment towards this hashtag is POSITIVE.



<u>Challenges faced with this project execution -</u>

- Instead of running the program file on AWS Lambda, I tried to spin up EC2 instance so that the server is maintained on EC2. I wanted to get all the tweets generated from the config file to be hosted on EC2, however, I was running into multiple errors while

integrating the project. I was unable to resolve the errors that were encountered for linking the .jar file to the EC2 instance. In the interest of time, I thought I should not try to resolve the errors and work only on Lambda instead and run the .jar file on Lambda

- Another problem I ran into was related to use of SNS / SQS. SNS is the Simple Notification Service of AWS whereas SQS is the Simple Queue Service. Initially I wanted to enable the SNS so that whenever the twitter file is uploaded into S3 bucket, a notification is sent explaining the details such as duration and file size. However, I was running into some configuration errors in that and a notification was not generated when the file is uploaded onto the S3 bucket. Instead, I used the SQS service which has a much better compatibility and the message was queued and were sent exactly once with the higher accuracy.

References:

https://github.com/cloudy-sentiment-analysis/CloudyMcCloudface
https://aws.amazon.com/blogs/big-data/building-a-near-real-time-discovery-platform-with-aws/
https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781785883231/9/ch09lvl1sec43/streaming-twitter-sentiment-analysis
https://cloudonaut.io/integrate-sqs-and-lambda-serverless-architecture-for-asynchronous-workloads/
https://sbstjn.com/serverless-sqs-worker-with-aws-lambda.html
https://aws.amazon.com/documentation/cloudtrail/
https://stackoverflow.com/questions/30748791/amazon-machine-learning-for-sentiment-analysis
https://www.researchgate.net/publication/303811249_A_Twitter_Sentiment_Analysis_for_Cloud_Providers_A_Case_Study_of_Azure_vs_AWS
https://github.com/bahuljain/Twitter-Analysis