THE BUSINESS OF CLOUD COMPUTING
FINAL PROJECT

TANUSHREE CHAUDHARY (1361677)

Guided by: Professor Tejpal Chadha

Part 1: Questions

Ques 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.

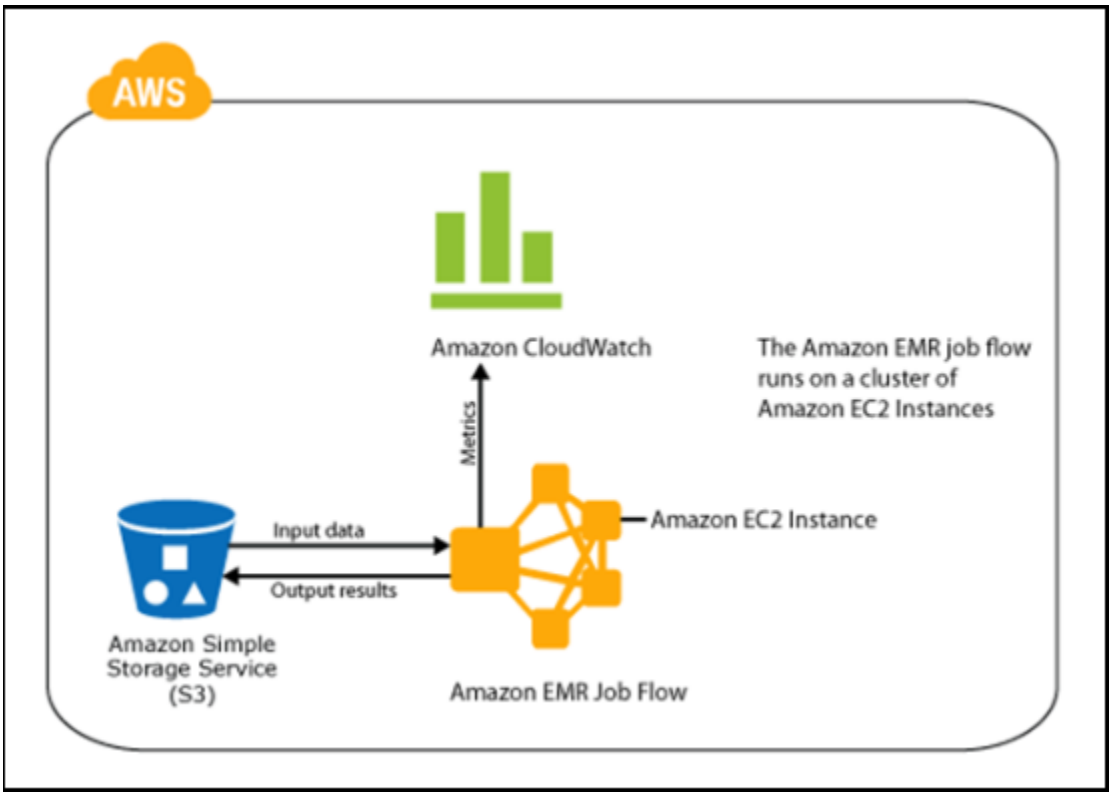
AWS:

PaaS Components AWS: Elastic Cache, Elastic Beanstalk(EBS)

SaaS Components AWS: Elastic MapReduce, Mobile Hub, Simple Notification Service(SNS)

Amazon Elastic MapReduce (EMR) is a managed framework such as Hadoop, Spark and Presto which can read and process unstructured data (files in strange formats). It can be used for web indexing, data warehousing, data analysis, so it typically provides us with capability to process large amount of data.

The EMR Job flow



Pricing: Amazon EMR is cost-efficient as you only pay for what you use. The hourly rate depends on the type of underlying instance used and the prices range from $0.011/hour to $0.27/hour.

Features & Benefits:

- With Amazon EMR, we can just focus on the data analysis and don't need to worry about the infrastructure, installing or running Hadoop, EMR takes care of it.
- EMR is elastic in the sense, we can set up or remove any number of clusters and as we launch clusters EMR adds up EC2 instances as per the requirement.

- EMR can be accessed using AWS console, CLI or SDKs.
- Amazon EMR is reliable as it keeps retrying the failed tasks and replaces the unhealthy instances by itself.
- With EMR, it is easy to start a cluster, or configure Hadoop or increase or decrease nodes as per the need. It also provides the facility to integrate with other AWS resources such as S3 and DynamoDB.
- EMR is secure as it configures security groups, provides option for restricted access to instances, and set up clusters in an isolated environment.

References: https://aws.amazon.com/emr/pricing/

https://www.packtpub.com/books/content/emr-architecture

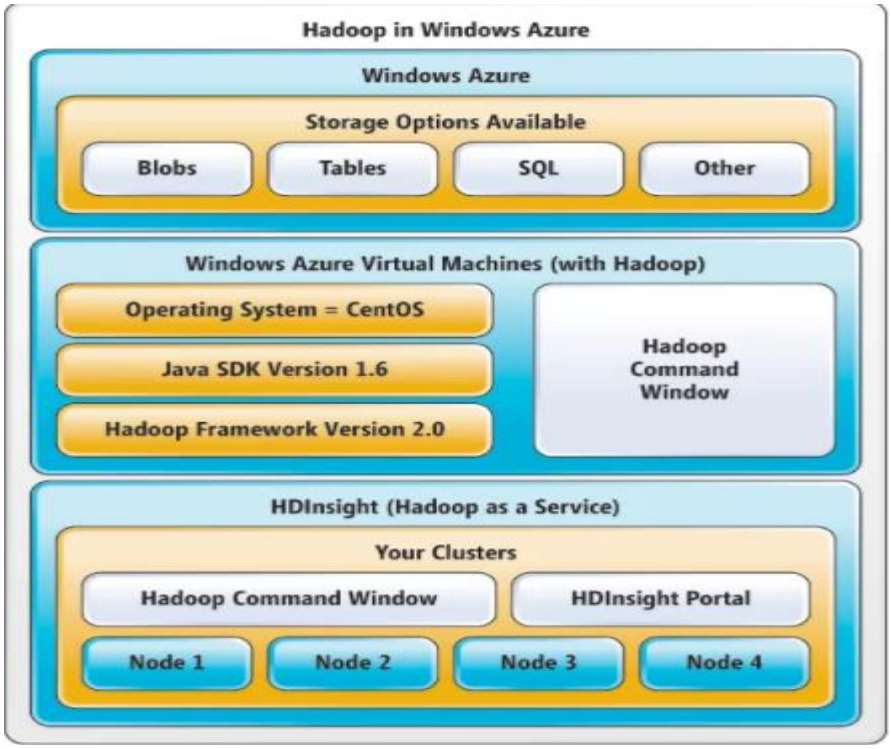
Microsoft Azure:

PaaS Components Azure: Azure HDInsight, Azure SQL databases, SQL reporting

SaaS Components Azure: Power BI, Log Analytics

Azure HDInsight is a fully managed cloud service operating on big data, which deploys and provisions Hadoop clusters in Azure cloud as per the customer requirement. Since it creates optimized clusters backed by 3 9's, the MapReduce framework, Hive, Spark and Pig have become more scalable and more reliable. HDInsight clusters have two master nodes which provides us with better task distribution and better availability.

Hadoop support in the Azure platform



Pricing: Azure HDInsight bills its users for the usage of the node for the life of the cluster, the per-hour billing rule. Billing starts and terminates with the cluster creation and deletion. Like AWS EMR, it also follows the pay as you go model. It gives you option to select between standard and premium clusters. Below are the pricing details for both:

	HADOOP, HBASE, STORM, SPARK	R-SERVER	KAFKA
Standard cluster	Base price/node-hour	Base price/node-hour + $0.08/core-hour	Base price/node-hour + $0/core-hour + Storage[1]
Premium cluster Preview	Base price/node-hour + $0.02/core-hour		

Features & Benefits:

- HDInsight provides an end-to-end SLA on the workload with automatic replication of data providing high availability.
- Since it is a fully managed service, it requires minimal maintenance and easy deployment.
- Control and flexibility of on-premise with automatic scaling of resources.
- From security and compliance point of view, HDInsight provides a single sign-on (SSO), and multi-factor authentication and helps us authorize users and groups with granular level access control.
- It provides developers and data scientists with choices of development environment like Visual Studio, IntelliJ, or Eclipse and later they can integrate the data using Jupyter notebook.

References: http://www.jamesserra.com/archive/2014/02/what-is-hdinsight/

https://azure.microsoft.com/en-us/services/hdinsight/

Ques 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:

Assumptions: On-demand pricing model, 4 EC2 Instances – t2.micro instance type, 1GiB memory deployed in region US-East Ohio. 1 Classic load Balancer for $0.025/hour. 2 DB instances of t2.micro type deployed in single AZ in region US-East Ohio.

Calculation:

4 EC2 Instances - Per hour pricing $0.0116

Cost for running 24*7 whole year= 0.0116*24*30*4*12 = $400.896

Region: US East (Ohio)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t2.nano	1	Variable	0.5	EBS Only	$0.0058 per Hour
t2.micro	1	Variable	1	EBS Only	$0.0116 per Hour
t2.small	1	Variable	2	EBS Only	$0.023 per Hour
t2.medium	2	Variable	4	EBS Only	$0.0464 per Hour

1 classic Load Balancer – Per hour pricing $0.025

Cost for running 24*7 whole year= 0.025*24*30*12 = $216

Classic Load Balancer Pricing by AWS Region

Americas | Europe | Asia Pacific

Americas

US East (N. Virginia and Ohio) and US West (Oregon)
$0.025 per Classic Load Balancer-hour (or partial hour)
$0.008 per GB of data processed by a Classic Load Balancer

2 DB Instances – Per hour pricing $0.017 and storage price for 2 db instances = $1.16 per month

Cost for running 24*7 whole year= 0.017*24*30*2*12 = $293.76 + 1.16*12 = $307.68

Region: US East (Ohio)

	Price Per Hour
Standard Instances - Current Generation	
db.m4.large	$0.175
db.m4.xlarge	$0.35
db.m4.2xlarge	$0.70
db.m4.4xlarge	$1.401
db.m4.10xlarge	$3.502
db.m4.16xlarge	$5.60
db.t2.micro	$0.017
db.t2.small	$0.034

Total cost for AWS = $400.896 + $216 + $307.68 = **$924.57**

To reduce costs in such a deployment, we can use reserved or spot instances. As there will be no changes in the number of instances and the application is going to run 24*7 for a whole year, reserving instances in advance can help the business to save money.

In Spot instances, we bid on unused EC2 capacity and the prices are determined by the supply and demand of spots. In comparison to on demand instances, these can help us save almost 80% of the price. Further, we can track the usage of the services and can optimize savings using cost explorer.

References: https://aws.amazon.com/rds/mysql/pricing/

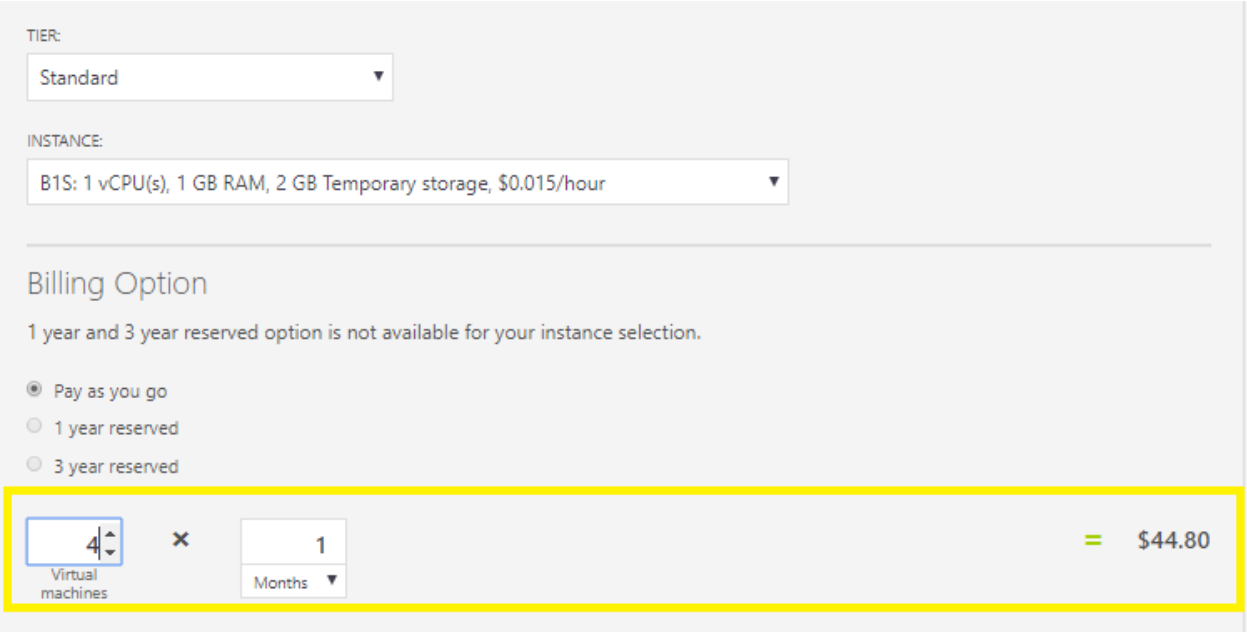
https://aws.amazon.com/blogs/big-data/strategies-for-reducing-your-amazon-emr-costs/

Microsoft Azure:

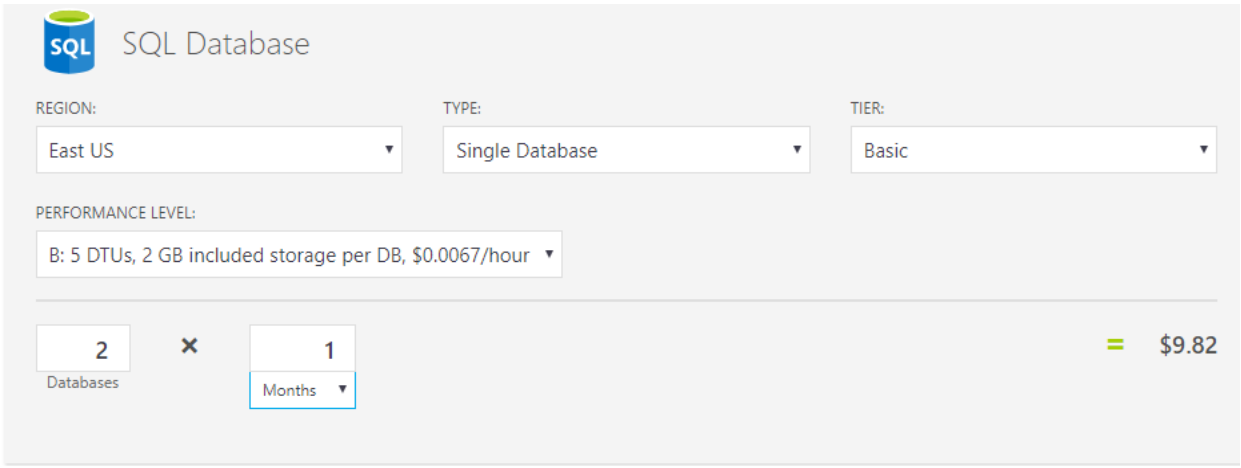
Assumptions: The billing option selected is Pay as you go. 4 Virtual machines deployed in US-east region on windows operating system with Standard tier for monthly usage. The instance type is B1S with 1GB RAM and 2GB temporary storage costing $0.015/hour. The SQL database is deployed in region US-east with basic tier with performance level B: 5 2GB storage included costing $0.0067. And the load balancer is free of charge for basic tier.

To run Azure cloud services for a whole year 24*7, the cost would be:

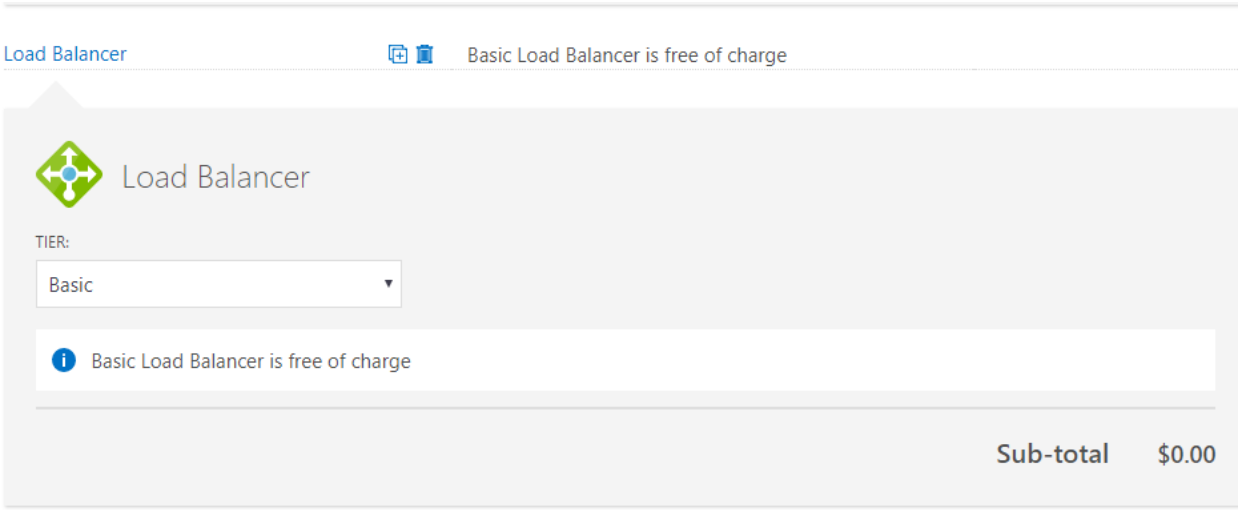
4 Virtual Machines cost $44.80* 12 months = $537.60



2 DB instances with basic tier costs, $9.82 * 12 months = $117.84



Since I am using basic tier, Load Balancer is free of use.

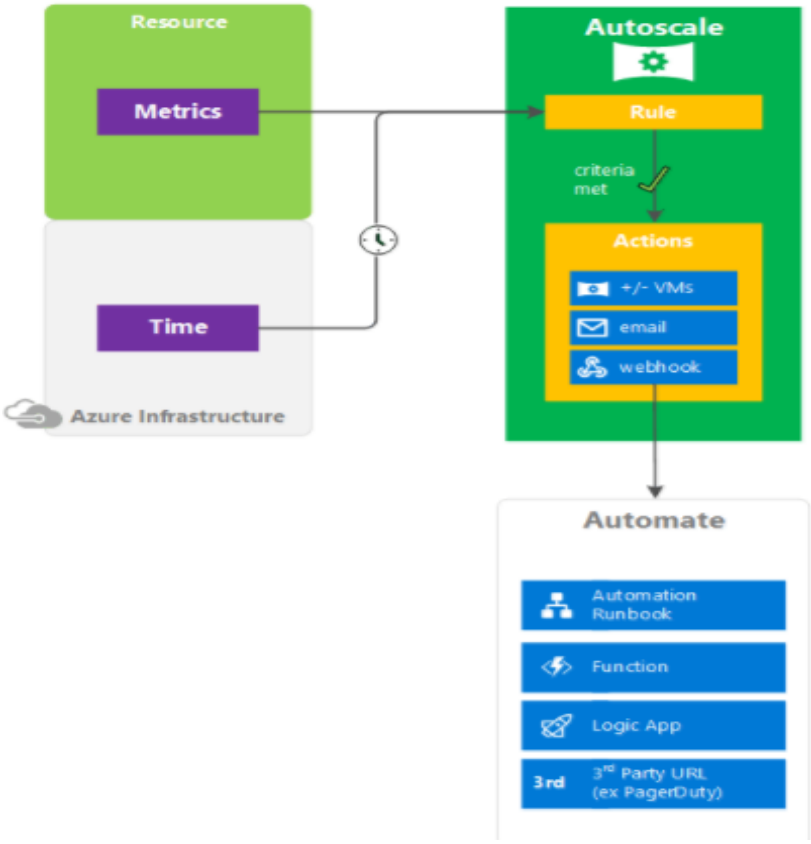


Total cost for Azure = $537.60 + $117.84 = **$655.44**

To reduce the costs in Azure, we can do the following:

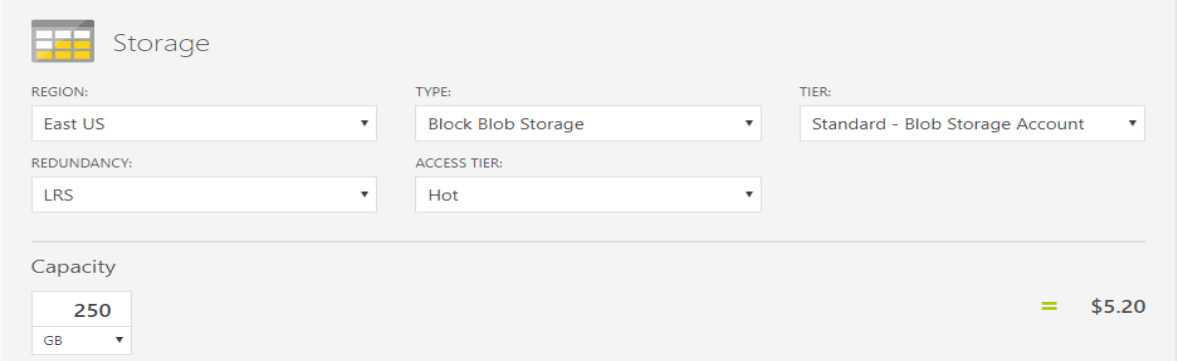
- To save on cost, we can minimize the compute hours by scaling up or down (horizontal scaling) the number of instances as per the service-level agreements. As the volume of traffic increases, the auto scaling group may add additional resources to meet the demand and removes those resources when the resources are no longer needed.

Conceptual diagram showing auto scaling process

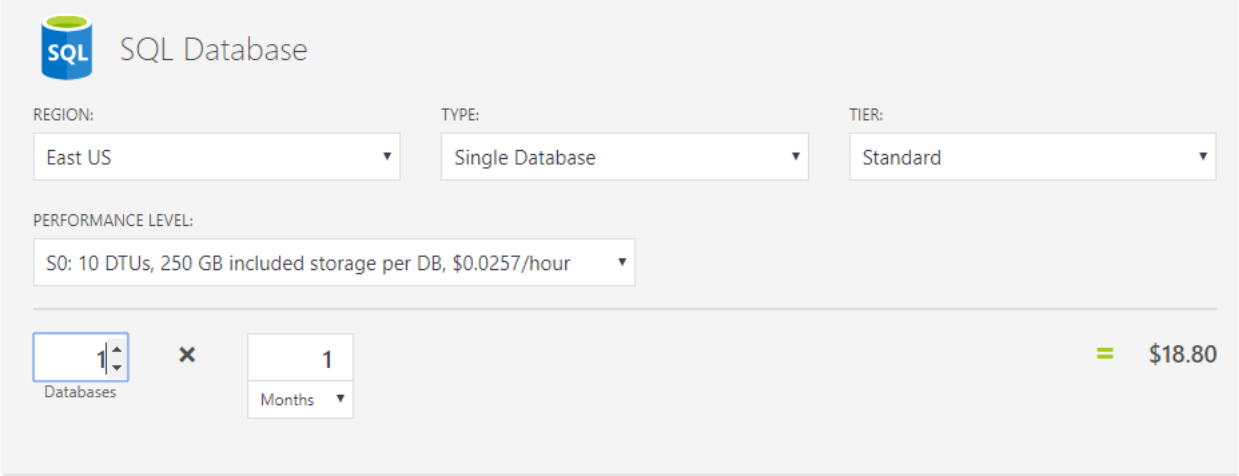


- Also, for DB instances, instead of using SQL database we can use blob storage which offers a cost-efficient solution for storing large items such as images, videos and text files. Costs for blob storage and SQL database is shown below:

Cost of using 250GB Blob Block Storage



Cost of using 250GB SQL Database



- The pricing for Azure changes region to region, so we can analyze the pricing and can deploy the application where the prices are low and can save some money. The prices vary in US regions also.

References: https://azure.microsoft.com/en-us/pricing/calculator/

https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-autoscale

---

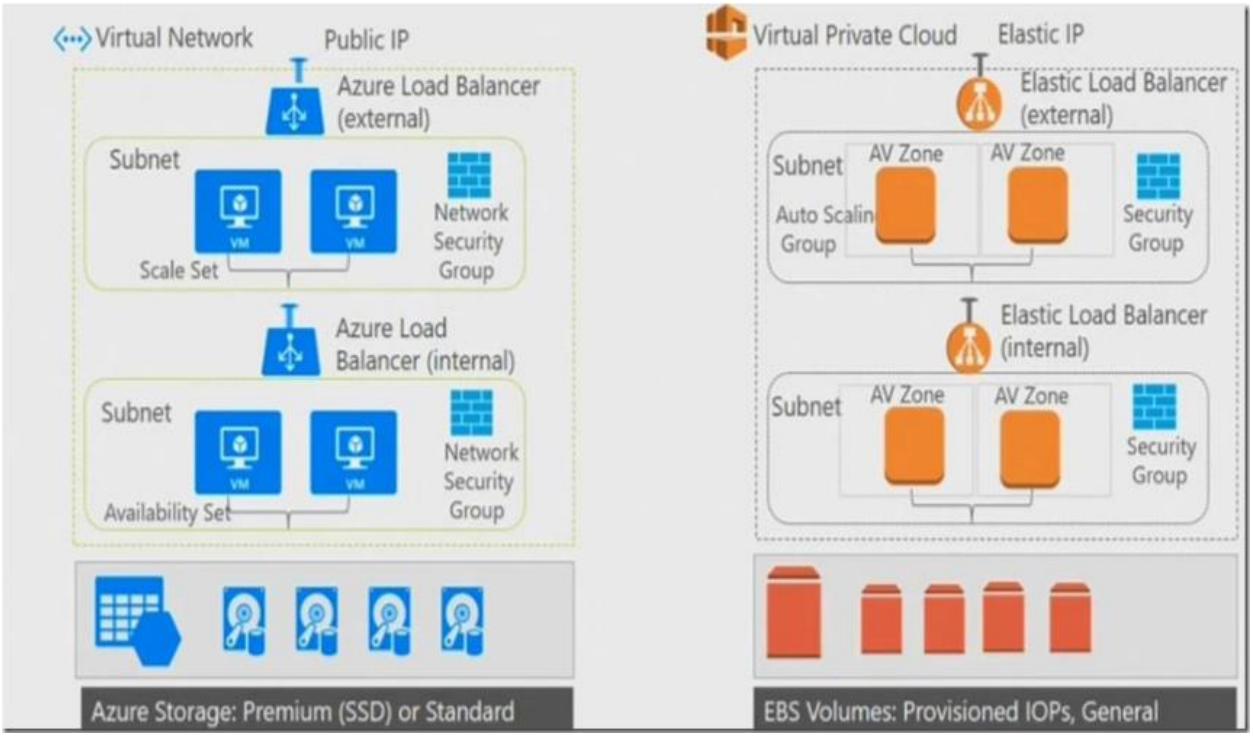
Ques 3: Compare the Security offering of the two Cloud vendors. Compare VPC, ACL and Security Groups that is provided by the two vendors. Which one is more secure and why? Give references, screen shots to support your answer.

AWS is the most well-known and the earliest cloud vendor in the industry providing services like EC2, RDS, VPC, Security groups and many more. **AWS is more secure** as it offers security at granular level by providing instance level protection and below we will be comparing the features of both the cloud vendors to see why AWS is better. Azure initially was just a PaaS provider and in 2013, it moved towards being IaaS provider with services like SQL databases, Virtual machines, VPC, DNS etc. Below are some security features offered by AWS and Azure:

Security Services Feature	AWS	Azure
Identity and Access Management	IAM	Active Directory
Key Management	KMS	Key Vault
Network	VPC	Virtual Network, ExpressRoute
Security Check	Trusted Advisor, AWS Inspector	Security Center
Storage Security	Data Encryption for S3	Storage Service Encryption (SSE)
Monitoring	CloudWatch	Azure Monitor, Application Insights
Logging	CloudWatch Logs, CloudTrail	Log Analytics, Security Event Logs
Compliance	CloudHSM	TrustCenter

VPC in AWS vs Azure: VPC provides user with a secured private network by encrypting data as it is transferred over the internet. The user is the sole controller of the environment and can perform services like creating subnets, configuring routing tables etc. AWS, being the pioneer provides more mature tools

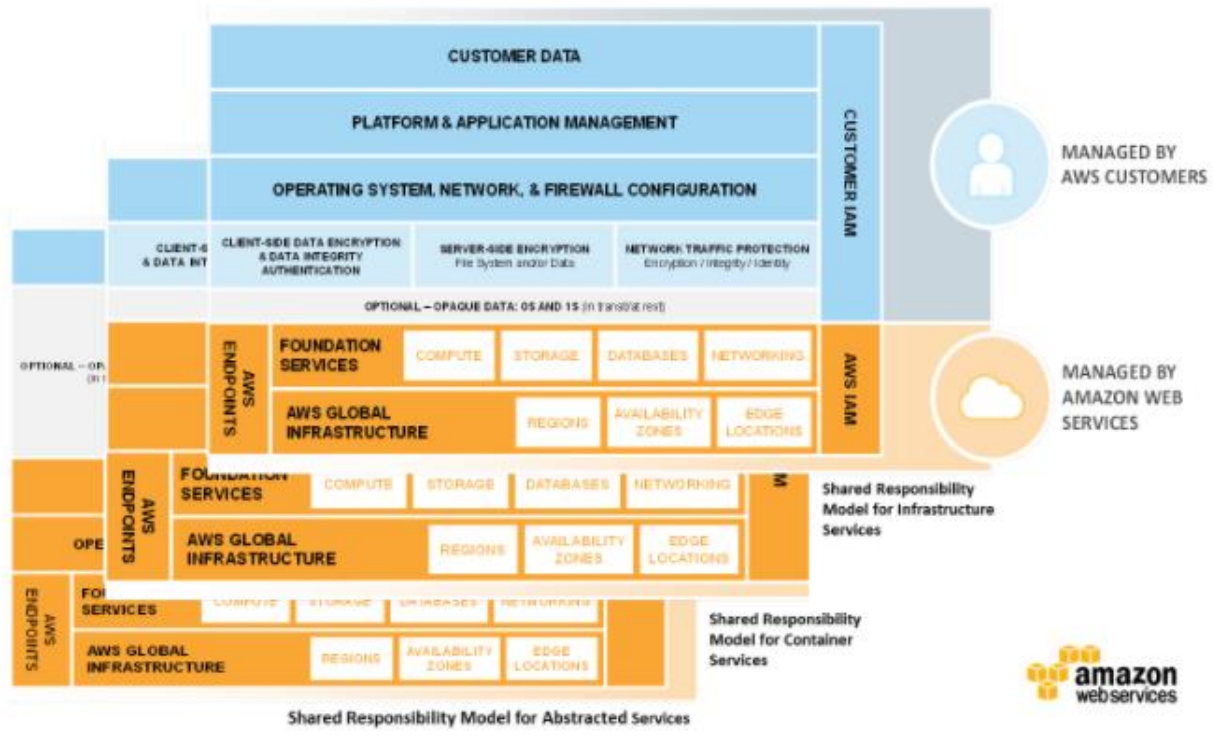
and provides with a functionality of automating the VPC architectures like VPC with a Single subnet, with public and private subnets and 2 more. Azure also provides us with the ability to create subnets using console or CLI, but it does not offer wizards to automate VPC architectures mentioned above. Also, AWS use routing tables to restrict routing while Azure uses firewalls or security groups as it does not have any route table feature.



ACL in AWS vs Azure: Access control list(ACL) provides us with a functionality of permitting or denying traffic to our networks. AWS and Azure both have NACLs as an added security measure on top of firewalls and security groups. With ACL in AWS, the EC2 instances in the subnet can receive http traffic, if we permit traffic to a subnet but if we deny the access, security groups will filter the unwanted traffic. In Azure, ACLs do not provide users with same control and flexibility and only secure endpoints. Azure recommends it users to use either NACLs or security groups, one at a time as they both provide the same functionality.

Security Groups in AWS vs Azure: In AWS**,** Security groups are the main reason for choosing Virtual networks as they guard your instances by filtering the incoming and outgoing traffic. Using security groups, we can easily specify inbound and outbound ports and the sources as well from which the ports can accept the traffic. AWS does not allow security groups to be across regions while Azure has 'Network Security group' as AWS's equivalent of Security group and it is only available for Regional Virtual Networks. Azure provides an add on feature by allowing the user to associate the VM instances and subnets to Network Security groups.

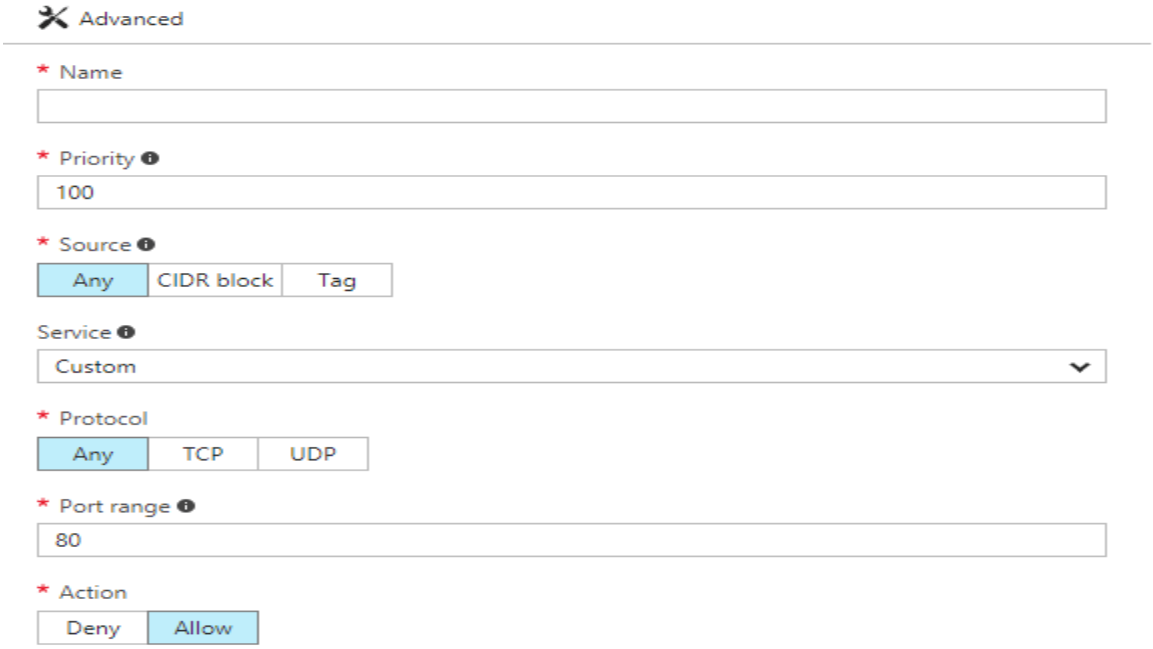
Security Model in AWS



Sharing Security Responsibility for AWS Services

In AWS, editing the inbound rules to allow/deny traffic on ports:

In Azure, controlling inbound and outbound traffic



References: https://8kmiles.com/blog/azure-virtual-network-vs-aws-virtual-private-cloud/
https://aws.amazon.com/blogs/security/new-whitepaper-aws-cloud-security-best-practices/

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

Docker creates software packages called containers that have all the dependencies like code, libraries, system tools required to run a software more quickly in any environment. Both AWS and azure have their own container services as EC2 Container Service (ECS) and EC2 Container Registry (ECR) for AWS and Azure Container Service(AKS). Below is the functional comparison between AWS and Azure container services.

Legend of icons used in comparison matrix tables:


Supported

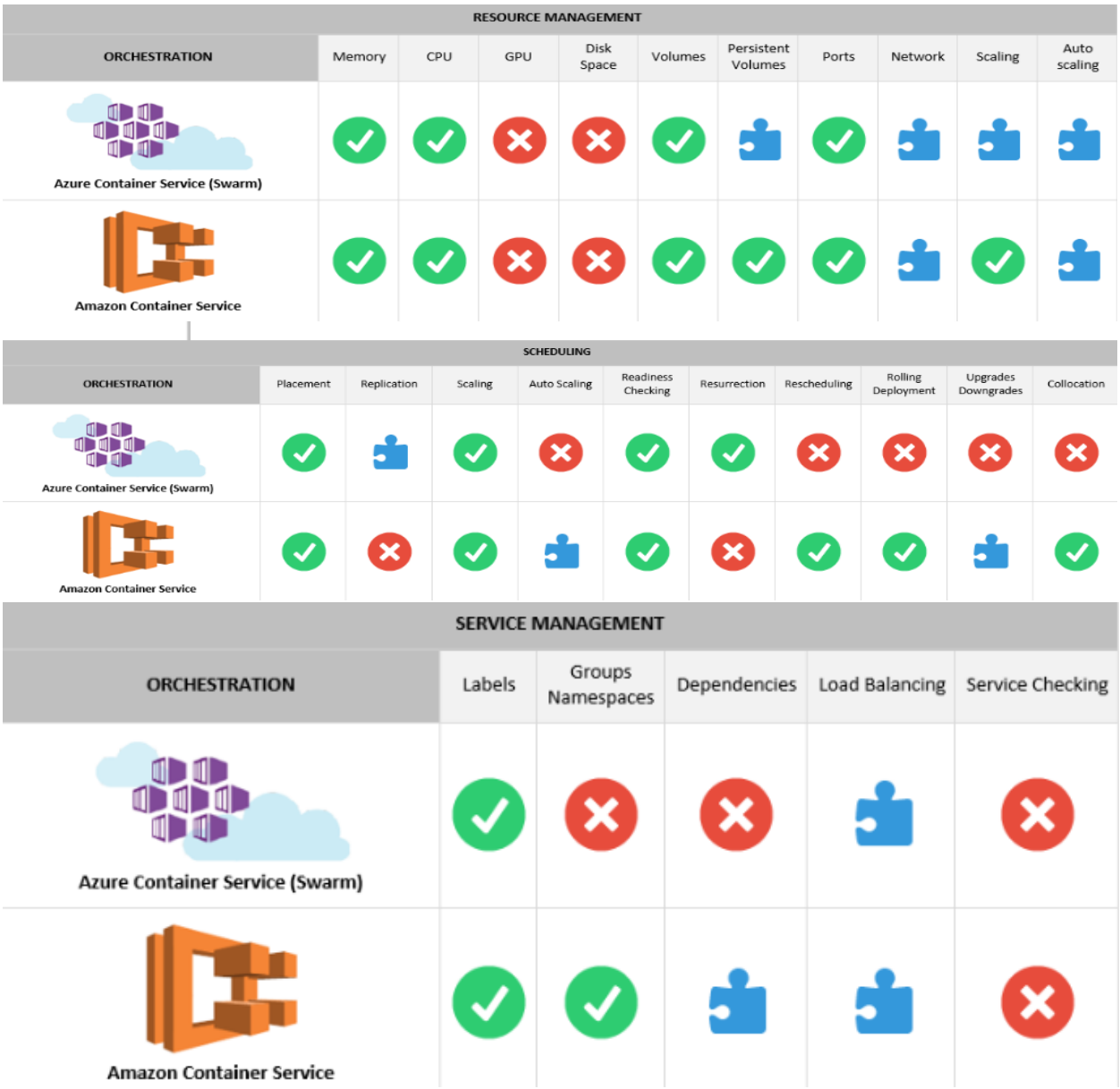
- Feature is supported in a matter of orchestration


Not supported

- Feature is unsupported in a matter of orchestration


Partially supported

- Feature support is covered by other service or component of Cloud Provider



ECS and AKS both supports portability of applications into any run time environment. Both provides users with features like security groups or firewalls, Elastic Load Balancers, Block Storage and Identity and Access Management. But in terms of reliability, AKS offers 99.9% performance for standard and even better with premium tier. When it comes to DevOps, ECS provides better integration as they have their own registries while Azure container service does not. ECS provides quality scalability because of AWS's highly scalable features like EC2 auto scaling groups where AKS provides fair scalability. AWS ECS has some operational issues like not able to monitor containers at granular levels.

Which is better?

Both have some pros and cons; the choice depends on the application needs a user has. So, there is nothing like one vendor is much better than the other as per my research. But surely, Amazon being the pioneer of cloud industry is certainly well established than Azure.

References: https://medium.com/devoops-and-universe/docker-swarm-at-aws-azure-vs-ce1b91a31eef
http://searchcloudcomputing.techtarget.com/opinion/Evaluating-Azure-Container-Service-vs-Google-and-AWS

Ques 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 automation tools provided by the two Cloud vendors.

Below is the list of some tools provided by Azure and AWS for automation:

Management Services & Options	Azure Resource Manager	Amazon CloudFormation
API Management	Azure API Management	Amazon API Gateway
Automation	Azure Automation	AWS OpsWorks
	Azure Batch Azure Service Bus	Amazon Simple Queue Service (SQS) Amazon Simple Workflow (SWF)
	Visual Studio	AWS CodeDeploy
	Azure Scheduler	None
	Azure Search	Amazon CloudSearch

AWS Command Line Interface provides us with ability to control and automate various AWS services through the command line using scripts. Azure on the other hand has Azure CLI and PowerShell which is also similar to CLI. For AWS CLI, there is lot of good documentation available on the internet about its functionalities whereas Azure lacks there. With Azure PowerShell, we can create managed platform services like Azure AppService which offers web hosting platform.

Automation tools provided by AWS:

- AWS CloudFormation: This provides developers and admins with sample templates or can also create their own templates to create, manage, provision and update the AWS resources as per the requirement.
- AWS CodeDeploy: This automates the deployments on an array of services like AWS EC2, or AWS Lambda, by managing itself the processes of handling the updating of software applications.
- AWS OpsWorks: This is a configuration management service that lets you use Chef and Puppet to automate server configuration and deployment where Chef and Puppet are automation platforms.

Automation tools provided by Azure:

- Azure Resource Manager: This enables us to deploy and organize app resources in a consistent state. With Resource manager we can control the access to the resources by defining roles and permissions.
- Azure Automation: It facilitates the users with automation of long manual work that requires repetition and even provides help them schedule those tasks automatically at regular intervals.
- Dynamics 365: This tool was released by Microsoft in fall 2017 to provide businesses with a cloud based combined product of ERP and CRM system.

References: https://aws.amazon.com
https://docs.microsoft.com/en-us/powershell/
http://www.tomsitpro.com/articles/azure-vs-aws-cloud-comparison,2-870-2.html

Ques 6: Research and compare 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.

Formula for calculating monthly uptime:

Monthly uptime percentage = $\frac{\text{total minutes per month} - \text{minutes downtime per month}}{\text{Total minutes in a month}}$

Formula for calculating downtime:

Monthly downtime percentage = $\frac{\text{Maximum Available Minutes - uptime}}{\text{Maximum Available Minutes}}$

AWS:

The SLA agreement for AWS and Azure says that they strive to achieve at least 99.99% of uptime and in case they fail to meet the commitment, they will issue service credits. Below is the service credit percentage to be issued in case of downtime using AWS EC2 and EBS services:

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.99% but equal to or greater than 99.0%	10%
Less than 99.0%	30%

There are websites like CloudHarmony which provides the uptime and downtimes for different cloud vendors on a real-time basis. As mentioned on CloudHarmony.com, below is the last 30 days availability schedule for AWS. As we can see, Amazon EC2 was up almost all the time except for US-west-2 region where availability was 99.9863% and 5.92 minutes of downtime, which is pretty good.

▾ **Amazon Web Services**

Service Name	Region	Status	30 Day Availability *1 block = 1 mins*	Outages	▲ Downtime
Amazon EC2	ca-central-1	↑	100%	0	None
Amazon EC2	us-east-1	↑	100%	0	None
Amazon EC2	us-east-2	↑	100%	0	None
Amazon EC2	us-west-1	↑	100%	0	None
Amazon EC2	us-west-2	↑	99.9863%	1	5.92 mins
Amazon Lightsail	us-east-1	↑	100%	0	None

Azure:

SLA for Azure guarantees 99.95% of uptime for virtual machines and 99.99% for storage and basic, standard and premium SQL database tiers. Below are the service credits to be issued when there is downtime:

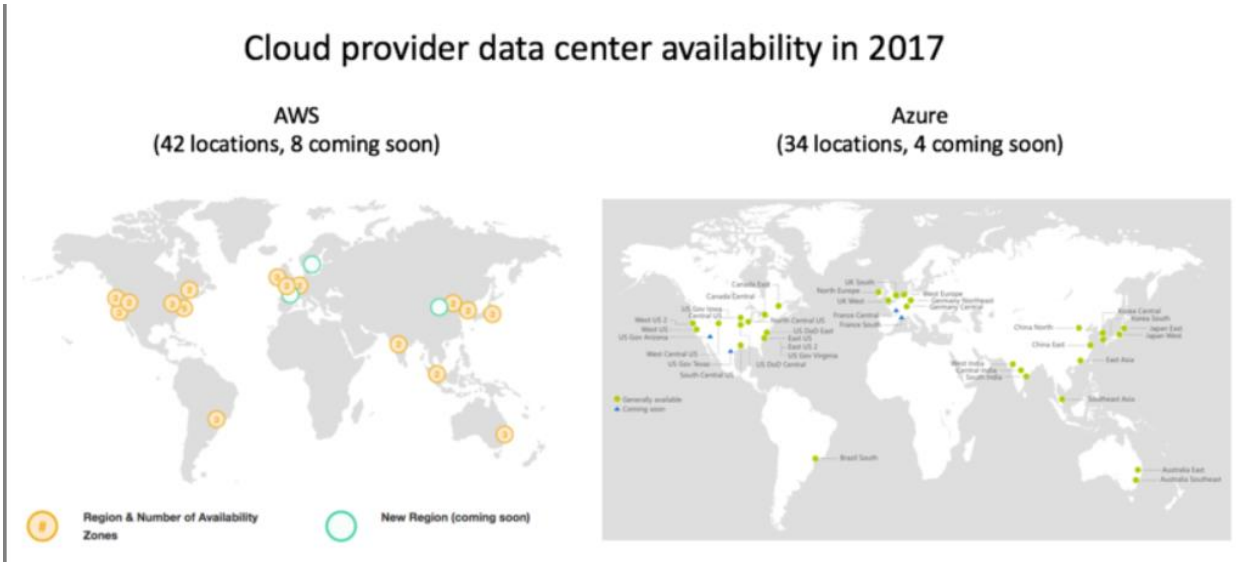
MONTHLY UPTIME PERCENTAGE	SERVICE CREDIT
< 99.9%	10%
< 99%	25%

Below is last 30 days availability schedule for Azure and it shows the Azure virtual machines were up almost all the time except for us-northcentral region where it was 99.7745% and 1.94 hours of downtime. This comparison of last 30 days clearly shows that AWS services have better uptime compared to Azure.

▾ **Microsoft Azure**

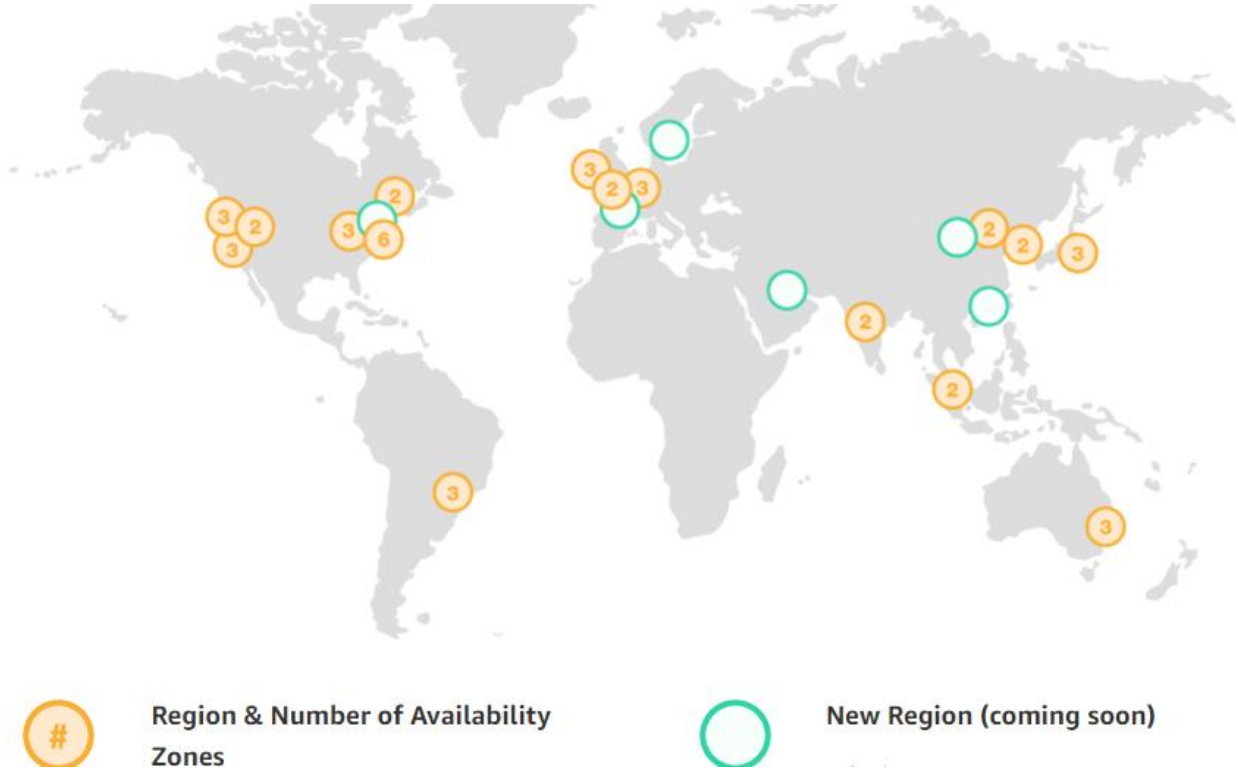
Service Name	Region	Status	30 Day Availability *1 block = 1 mins*	Outages	▲ Downtime
Microsoft Azure Virtual Machines	canada-central	↑	100%	0	None
Microsoft Azure Virtual Machines	canada-east	↑	100%	0	None
Microsoft Azure Virtual Machines	us-central	↑	100%	0	None
Microsoft Azure Virtual Machines	us-east	↑	100%	0	None
Microsoft Azure Virtual Machines	us-east2	↑	100%	0	None
Microsoft Azure Virtual Machines	us-northcentral	↑	99.7745%	3	1.94 hours
Microsoft Azure Virtual Machines	us-southcentral	↑	100%	0	None
Microsoft Azure Virtual Machines	us-west	↑	100%	0	None
Microsoft Azure Virtual Machines	us-west2	↑	100%	0	None
Microsoft Azure Virtual Machines	us-westcentral	↑	100%	1	25.52 mins

Below is the graph about data centers around the world for AWS and Azure. AWS currently has 42 locations and 8 more are coming while for Azure, there are data centers at 34 locations and 4 more are coming soon as per reports in 2017. Having regions, AZs and data centers around the world, helps these cloud vendors to provide businesses with replication of data and distribution of workloads across different locations as per the demand. More regions provide users with less delay and high availability.

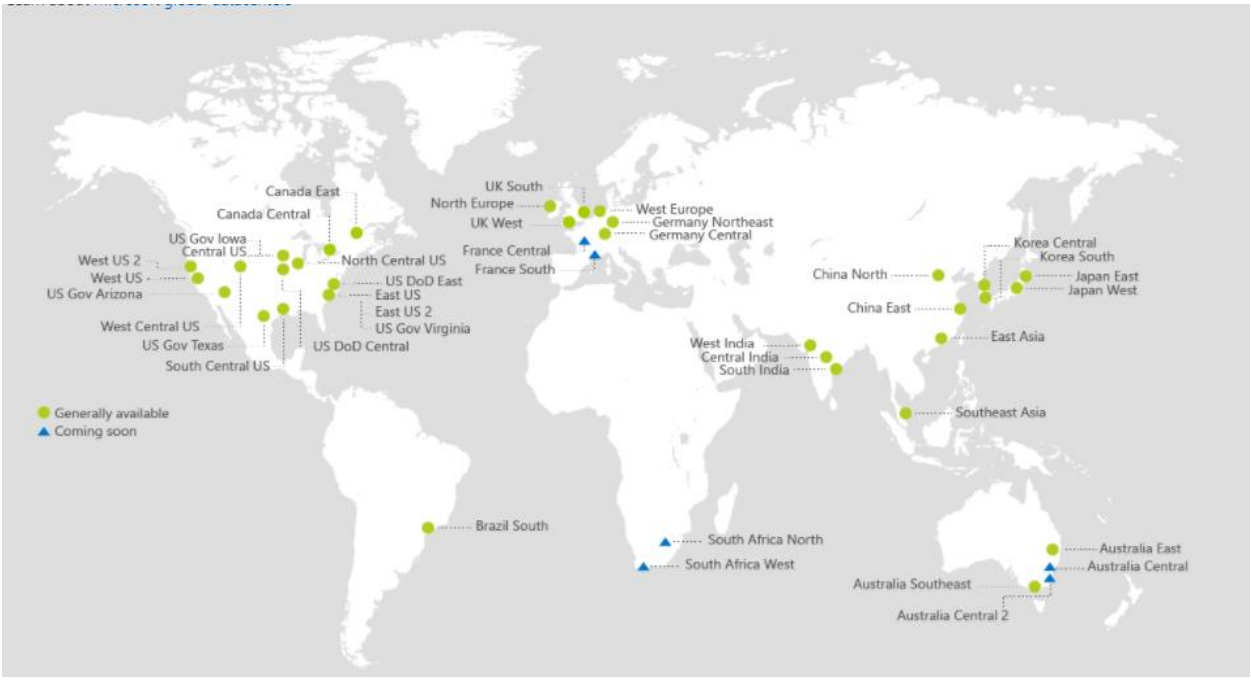


Detailed information about data centers:

AWS



Azure



References: https://cloudharmony.com/status-of-compute-in-america_north-group-provider
https://aws.amazon.com/ec2/sla/
https://azure.microsoft.com/en-us/support/legal/sla/summary/

https://community.spiceworks.com/cloud/articles/2496-aws-vs-azure-in-2017-iaas-similarities-and-differences

---

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

AWS and Azure both have auto scaling groups that scales up or down the resources to meet the demands as per the requirement and improve the application availability. It lets you provide maximum, minimum and desired number of instances. Since, it provisions and cuts down the VMs as per the demand, it lessens the cost and at the same time keep up the performance.

AWS:

Pricing: in AWS is enabled by Amazon CloudWatch and is operated at no extra price. Both EC2 and CloudWatch are billed separately and straight away charges for the whole hour even if used for partial hour. Below are the pricing details for CloudWatch:

Region: US East (Ohio)

Amazon CloudWatch Dashboards
- $3.00 per dashboard per month

Detailed Monitoring for Amazon EC2 Instances
- $2.10 down to $0.14 per instance per month at 1-minute frequency******

Amazon CloudWatch Custom Metrics
- $0.30 per metric per month for the first 10,000 metrics
- $0.10 per metric per month for the next 240,000 metrics
- $0.05 per metric per month for the next 750,000 metrics
- $0.02 per metric per month for metrics over 1,000,000

Amazon CloudWatch Alarms
- $0.10 per alarm per month
- $0.30 per high-resolution alarm per month

Amazon CloudWatch API Requests
- $0.01 per 1,000 GetMetricStatistics, ListMetrics, PutMetricData, GetDashboard, ListDashboards, PutDashboard and DeleteDashboards requests

Amazon CloudWatch Logs*
- $0.50 per GB ingested**
- $0.03 per GB archived per month***
- Data Transfer OUT from CloudWatch Logs is priced equivalent to the "Data Transfer OUT from Amazon EC2 To" and "Data Transfer OUT from Amazon EC2 to Internet" tables on the EC2 Pricing Page.

Amazon CloudWatch Events - Custom Events****
- $1.00 per million custom events generated*****

Features:

- Auto scaling groups ensures the EC2 instances availability by always maintaining the desired number of instances and scaling up or down the resources as the demand changes.
- Monitors the health of instances, and makes sure the traffic is sent to healthy instances only.

- Balances the EC2 instances across the availability zones providing users with less downtime.

Azure:

Pricing: Unlike AWS, Azure does not require us to pay additionally for auto scaling groups, you just have to pay for Azure CloudWatch and the number of instances used. Below are the pricing details about basic deployment:

A0-4 Basic

A Basic is an economical option for development workloads, test servers, build servers, code repositories, low-traffic websites and web applications, micro services, early product experiments, and small databases.

ADD TO ESTIMATE	INSTANCE	vCPU	RAM	TEMPORARY STORAGE	PAY AS YOU GO	1 YEAR RESERVED (% SAVINGS)	3 YEAR RESERVED (% SAVINGS)	3 YEAR RESERVED WITH AZURE HYBRID BENEFIT (% SAVINGS)
⊕	A0	1	0.75 GiB	20 GiB	~$13.18/month	- -	- -	- -
⊕	A1	1	1.75 GiB	40 GiB	~$23.43/month	- -	- -	- -
⊕	A2	2	3.50 GiB	60 GiB	~$97.36/month	- -	- -	- -
⊕	A3	4	7.00 GiB	120 GiB	~$216.68/month	- -	- -	- -
⊕	A4	8	14.00 GiB	240 GiB	~$433.35/month	- -	- -	- -

Reserved VM Instances are not available for the A series.

Features:

- Azure requires you to pre-provision the number of Virtual Machines and then auto scale turns them on or off as per the demand.
- Allows you to anticipate the load at different time intervals and scale the VMs at those times.
- Monitors key performance metrics and if there is some inconsistency or change like drop in response time, it informs the user.

References: https://azure.microsoft.com/en-us/features/autoscale/
https://azure.microsoft.com/en-us/pricing/details/virtual-machine-scale-sets/windows/
https://aws.amazon.com/cloudwatch/pricing/

Ques 8: How does the AWS S3 compare or Object Store compare 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?

AWS:

Features in AWS: AWS provides Simple Storage System(S3) as object storage which allows to store and fetch the data anywhere anytime with 99.999999999% durability. In S3, data is automatically distributed among geographically separated regions and it gives you option to replicate the data to any AWS region. It provides security and compliance capabilities in the form of encryption of data. It facilitates the user to run analysis on large amount of data (big data) without moving it to any other data analytics system. Amazon S3 also allows you to create versioning of the stored data for backup and recovery stand point. S3 automatically deletes the objects after a time interval to save on space.

Pricing: AWS S3 gets you started free with 5GB of S3 storage. After that you pay for what you use, cost depends on the request type and the quantity of requests. S3 does provide us with two options one is standard storage and other one reduced redundancy to lower down the costs. Below are the pricing details:

	Standard Storage	Standard - Infrequent Access Storage †	Glacier Storage
First 50 TB / month	$0.023 per GB	$0.0125 per GB	$0.004 per GB
Next 450 TB / month	$0.022 per GB	$0.0125 per GB	$0.004 per GB
Over 500 TB / month	$0.021 per GB	$0.0125 per GB	$0.004 per GB

Azure:

Features in Azure: Azure provides Azure Blob Storage service for object storage. Blob storage allows you to import or export big data to and from Azure but does not clear the storage automatically in some time. Like AWS, it also provides data security and denies unauthorized access to the data by enabling secure policies. It does not facilitate with data encryption. It gives data redundancy options like zone redundant storage, locally redundant storage, geographically redundant storage etc.

Pricing: Azure also does not require any upfront costs**.** For Azure, the storage prices vary based on the tier such as Hot, cool or archive and also the type of redundancy selected and of course the quantity of data stored. Below are the pricing details for each tier:

	HOT	COOL	ARCHIVE(PREVIEW)
First 50 TB / Month	$0.0208	$0.0152	N/A
Next 450 TB / Month	$0.02	$0.0152	N/A
Over 500 TB / Month	$0.0192	$0.0152	N/A

Comparison between AWS and Azure object storage:

Object Storage - Overview

	AWS	Azure	Google
Service Name	S3	Azure Storage (Blobs)	Google Cloud Storage
Availability SLA	99.95%	99.99%	99.95%
Hot	S3 Standard	Hot Blob Storage	GCS
Cool	S3 Standard – Infrequent Access	Cool Blob Storage	GCS Nearline
Cold (Archival)	Glacier	Use Cool Blob Storage	GCS Coldline
# Object Limits	Unlimited	Unlimited	Unlimited
Size Limit	5 TB/Object	500 TB/account	5TB per object

Based on the analysis, I would prefer using Amazon's S3 bucket for object storage though Azure has high availability but still Aws S3 is equipped with more features. Also, Amazon has been the first in this market. Features like hosting static websites, providing fine grain control, bucket versioning have an edge over Azure's blob storage.

References:https://www.cloudberrylab.com/blog/amazon-s3-azure-and-google-cloud-prices-compare/https://azure.microsoft.com/en-us/pricing/details/storage/blobs/
https://www.networkworld.com/article/3191520/cloud-computing/deep-dive-on-aws-vs-azure-vs-google-cloud-storage-options.html

---

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

Amazon's Elastic Compute Cloud(EC2) and Azure's Virtual Machines(VMs) are both the virtual servers that facilitates a cloud user with scalable computing on demand. Users have the flexibility to scale up or down the resources (auto scaling groups) as per the need and just pay for what they use.

Features in EC2 and VMs: Both EC2 and VMs are scalable via auto scaling. AWS allows both auto-scale and resizing an EC2 instance when the instance is in stopped state while in Azure, instances can be easily resized without having to stop the instance but there will be some change in the tier specifications. Backup facilities in EC2 and VMs is all the same, both provides image creation, snapshots and file backups as different ways to take a backup.
As every cloud consumer would like its application to be available all the time, so uptime is also an important factor. AWS EC2 SLA gives its users 10% discount if the Virtual machine was unavailable for more than 0.05% time of the month, Azure also provides the same service credit for 0.05% downtime. But Amazon provides 30% discount if the virtual machine was unavailable for more than 1% time of the month while Azure's discount offering for this is 25%. Below is the detailed difference between the three main cloud compute engines:

	Amazon EC2	Google CE	Microsoft Azure VM
Number of instance templates available	39	18	40
GPU acceleration	Yes	No	No
Custom instance creation feature	No	Yes	No
CPU Limits	1 - 40	1 Shared - 32 dedicated CPU	1 - 32 CPU
Memory Limits	0,5 - 244 GB	0,6 — 208 GB	0,75 — 448 GB
Temporary Storage Limits	Up to 48 TB (Multiple Disks)	3 TB	2 TB
Network features supported	CDN, Direct connection, DNS, Load Balancing, Virtual private cloud network, VPN Gateway		

Below are the instance types, memory and storage offered by AWS EC2 and Azure's Virtual machines:

Resource Type (us-east, Linux)	AWS Instance	AWS Memory	AWS Storage	Azure Instance	Azure Memory	Azure Storage
Standard 2 vCPU w SSD	**m3.large**	8	32	**D2 v2**	7	100
Highmem 2 vCPU w SSD	**r3.large**	15	32	**D11 v2**	14	100
Highcpu 2 vCPU w SSD	**c3.large**	3.75	32	**F2**	4	32
Standard 2 vCPU no SSD	**m4.large**	8	0	**D2 v2**	7	100
Highmem 2 vCPU no SSD	**r4.large**	15.25	0	**D11 v2**	14	100
Highcpu 2 vCPU no SSD	**c4.large**	3.75	0	**F2**	4	32

As of Dec 2, 2016

Pricing: AWS EC2 bills its users for one full hour starting from the first minute of usage while Azure bills 1/60th of the hourly cost for each minute the service is used. The prices for compute engine depends on your use case whether you are going for Reserved, on demand or spot instances. AWS gives discounts when you go for reserved instances by paying upfront for 1-3 years and you have a defined usage for that time interval. Below is the on-demand pricing for AWS and Azure:

Resource Type (us-east, Linux)	AWS Instance	Azure Instance	Google Instance	AWS OD Hourly	Azure OD Hourly
Standard 2 vCPU w SSD	m3.large	D2 v2	n1-standard-2	$0.133	$0.114
Highmem 2 vCPU w SSD	r3.large	D11 v2	n1-highmem-2	$0.166	$0.149
Highcpu 2 vCPU w SSD	c3.large	F2	n1-highcpu-2	$0.105	$0.099
Standard 2 vCPU no SSD	m4.large	D2 v2	n1-standard-2	$0.108	$0.114
Highmem 2 vCPU no SSD	r4.large	D11 v2	n1-highmem-2	$0.133	$0.149
Highcpu 2 vCPU no SSD	c4.large	F2	n1-highcpu-2	$0.105	$0.099

As of Dec 2, 2016

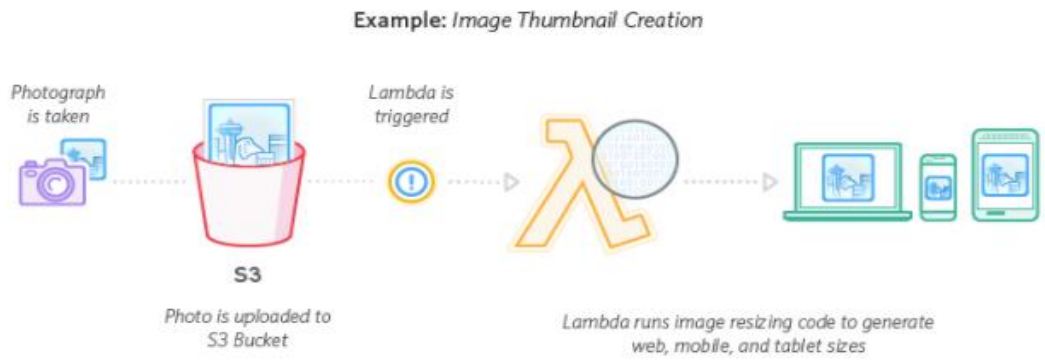
When it comes to preference, every cloud vendor has its own services and their benefits. The bottom line is, it totally depends on a particular use case. Though Azure's pricing is a bit low than AWS but still AWS provides us with an array of better features like multiple memory combinations. So, AWS being more established and offering better uptime because of more data centers is definitely the choice.

References: https://www.rightscale.com/blog/cloud-cost-analysis/aws-vs-azure-vs-google-cloud-pricing-compute-instances
https://www.cloudberrylab.com/blog/azure-vm-vs-amazon-ec2-vs-google-ce-cloud-computing-comparison/

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

AWS:

AWS Lambda, one of the most publicized service introduced by Amazon is definitely a really cool feature of AWS. It allows you to run the code for any application without having you to administrate the servers. You just upload the code and Lambda automatically manages the running and scaling of the application. AWS Lambda executes a particular code whenever there is a trigger. Lambda by making the use of trigger, makes it convenient to build and manage services where we need the response fast. For example, if we have a business application that has its database located on AWS. By using lambda, we can update the business application whenever there is an update in the database. So, we can set the trigger action based on the database update and do real time file processing.



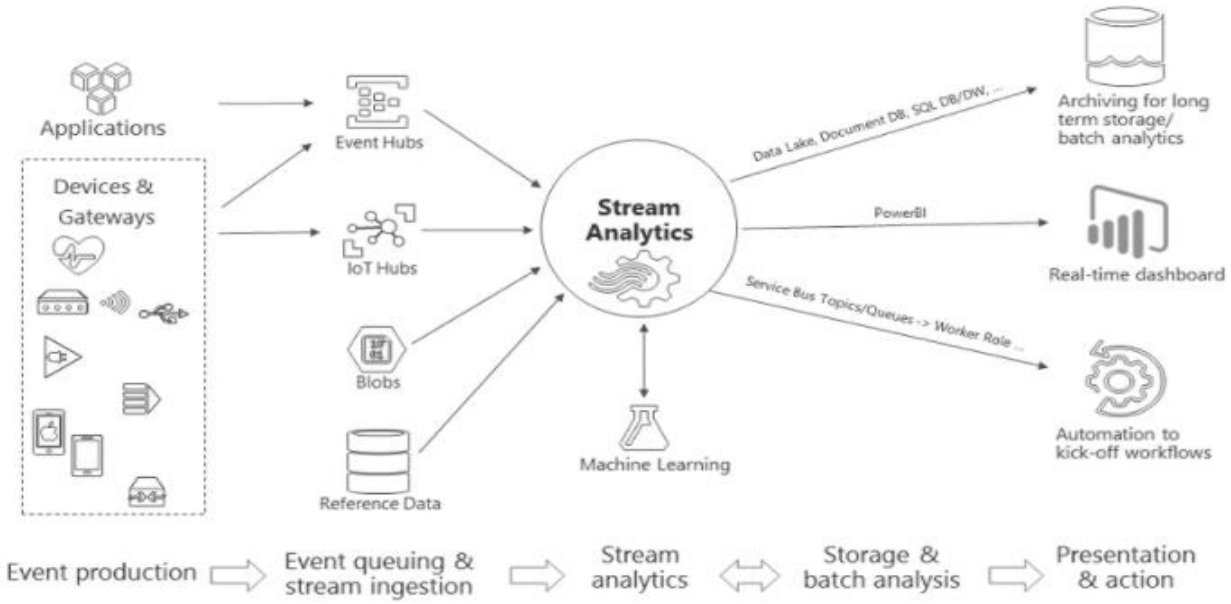
Some more features are:

- Real time Stream processing uses AWS Lambda and Amazon Kinesis for tracking application activity, indexing, processing transaction, cleaning data, social media analysis and many more.
- Extract, Transform, Load using Lambda
- Can build serverless backends which handles mobile or web or IOT.
- Using AWS lambda along with other AWS services, can develop web applications with automatic scaling.

Pricing: AWS Lambda lets you to pay for every 100ms the code is executed and also the number of times the code is executed. It provides with sub-second metering and you don't need to pay anything when the code is not running. The free tier includes free first 1 million requests per month and the first 400,000 GB-seconds of compute time. Hence, it is really cost-efficient.

Azure:

Azure Stream Analytics, performs real time analytics on the streaming data coming from websites or sensors or any social media platform. Using Stream analytics, we can extract information, analyze the trends and patterns it is showing and then use these trends to trigger alerts or store for later use. Stream Analytics along with Azure's Event Hubs allows you to process large amount of data and it can handle 1GB of incoming data per second. Since it is a managed service by Azure, it is reliable in the sense it ensures business continuity and in case of a failover event, it has built in recovery service.
Below diagram shows how Stream analytics work:



Some examples include:

- Real-time fraud detection and identifying the protection measures
- Analyzing the real-time stock data and creating alerts
- Analysis of a user's web clickstream

Pricing: Azure Stream Analytics optimized from a cost standpoint as it bills a user as per the streaming unit's usage required to analyze the data. The usage is derived as per the compute power used and the quantity of events processed for the job to be done. Below is the pricing snapshot for Stream Analytics standard streaming unit:

Pricing details

Azure Stream Analytics is priced by the number of streaming units required to process the data into the service.

Standard Streaming Unit

USAGE	PRICE
Streaming Unit	$0.11/hour

References**:** https://aws.amazon.com/lambda/?p=tile
https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction

Part 2: BUSINESS SECTION

Migrate a legacy Application (POS (Point of Sale) System at a Starbucks) to Cloud. Mention the Challenges, Opportunities and mechanics. Also, imagine now that the entire software and backend is in the Cloud and Starbucks can take orders on an iPad or phone. What would be the Pros and Cons of such a Cloud deployment.

Industry Overview

Starbucks is an American coffeehouse chain, originally founded in Seattle, Washington in 1971. It has a strong market presence around the world with 23,768 locations worldwide as of today. Starbucks has been strategic in placing its stores ate high visibility and high traffic areas. Though the prices for its products are high as compared to a normal coffee but it has introduced loyalty based programs like Starbucks Rewards program to motivate and reward its customers. When it comes to technology, it has provided its customers with mobile application to order online and pick up the order. For every business, it is important to stay ahead of the competition, especially for IT departments. The legacy systems that have majorly contributed in the success of the business, they might be costing more now and might be difficult to manage and keep up with the compliance. According to a study, an average of 80% budget of IT department is spent on maintaining the legacy systems. So, now we will be discussing the legacy application used by the well-known coffee chain “Starbucks”, challenges faced with the legacy system, solutions provided by the cloud infrastructure and if there are any risks associated in moving to the cloud. In the end, we will be analyzing do Benefits outweigh the risks? Should we migrate to cloud or not?

Legacy Application

Over 7000 Starbucks outlets in United States have been using Point of Sale(POS) system provided by Square since 2012. The POS system offered by Square allows its customers to process the transactions using mobile platforms like Android, iPhone, or iPad by downloading the Starbucks mobile and loyalty app. This mobile payment app is daily used by almost 10 million customers in the US. Starbucks has also been associated with MICROS to use Simphony POS system across many locations.

Challenges faced using the Legacy system

- **Less Flexibility:** Starbucks usually has peak time in the morning and evenings when the baristas must process lot of transactions using the POS. So, basically the main demand of the system is during the peak hours and the system is less engaged in non-peak hours. But legacy POS systems do not offer flexibility to scale up or down the resource usage as per the demand and, yet the compute power and other resources are wasted.
- **Usage of traditional computer systems:** Starbucks process the day-to-day transactions using computers that are supported by the POS systems. With the increasing customers, they have to process a large amount of data which exceeds the limits of a traditional system RAM and memory and storage. Hence, the speed is affected and as a result, long queues!
- **Unavoidable downtime:** Since all transactions are done using hardware and data is stored locally at the machines. In case, there is a disaster or malfunction in the system, the whole system will have to be shut down. This way customers will get affected and they might move to other coffee shops and Starbucks can lose the profit.

  Example: Starbucks POS outage: April 24th, 2015
  On April 24th, 2015, Starbucks experienced a POS outage across United States and Canada, as a result the coffee house was not able to process customers transactions and had to close the stores early or give away free coffees. The next day, Starbucks assured that it was an internal outage and not a security or data breach.

- **Security and Compliance:** The POS system stores the financial data of the customers doing the transaction, this helps Starbucks to access their records next time faster. In case, the system is hacked by someone, the security of customers data is breached, and Starbucks will have to pay for it and also will lose its valuable customers. Compliance standards such as HIPAA can lay serious penalties due to security holes in the system.
- **IT Personnel cost:** Starbucks is paying a lot of money to the IT department employees who are responsible for maintaining the legacy systems. In case of system failure, there could be overtime costs to meet the deadlines.
- **Maintenance cost:** The old legacy systems require more time and money to get updated and that requirement could arise at the business hours, which can be a problem for the customers. The sales would suffer, and the profits will get affected. So, in order to avoid this scenario proper maintenance scheduling arrangements are to be done in advance. Also, the initial installation cost is an add-on for the hardware system.

- **No Disaster Recovery:** Legacy systems are not very technologically advanced, so there is no procedure of applying security patches for critical data. As Starbucks process credit cards, strict security measures should be taken. Backup and disaster recovery becomes a challenge in legacy systems. As everything is there on the system, you cannot safeguard the customer data in case of a failover event and will always be at a risk of losing data.

Solution provided by the cloud

To solve the problems listed above, to decide whether we should migrate the legacy application at Starbucks to Cloud or not? let's analyze the solutions provided by the cloud Vendor Amazon Web Services(AWS).

- **Elasticity:** AWS solution to the problem 'less flexibility' is elasticity. It provides its users with automatic resource provisioning or decreasing using auto scaling groups as per the demand. Starbucks has some peak hours, so we can easily set the auto scaling rules to add the instance when the demand spikes and terminate them when demand is low.
- **Fast system performance:** Since the data is stored on the local machine, in times of increased loads the system can get slow. This can result in affected performance and long queues. AWS provides its users with faster retrieval of data from its storage units like S3. There will be no more problem of memory and storage and the authorized company employees can access the data from cloud anytime anywhere.
- **Less Downtime:** The SLA agreement for AWS says that they strive to achieve at least 99.99% of uptime which means only 4 minutes 19 seconds of downtime. In case they fail to meet the commitment, they will issue service credits up to 30% for the business loss.
- **More Secure:** In a legacy system, just the closed door is the main defense to secure the equipment and the business data. While in cloud vendors like AWS, there are increased security measures like firewalls and security groups through which we can filter the incoming traffic to our application and keep out the bad guys from accessing our system and keep the financial records safe. The security rules in the cloud follow regulatory standards like HIPAA and offers security against major attacks like DDoS attacks.
- **Minimal maintenance and Personnel cost:** As everything is on cloud and we don't have to manage any physical servers or storage on system, so there will be minimal maintenance and IT personnel cost to manage the IT. AWS will be taking care of the updates and make sure the servers are running.
- **Redundancy:** AWS has a great disaster recovery option. It offers us the Multi AZ deployment, which allows you to deploy your application in multiple availability zones. So, if there is a failover event in one of the zone, there is no single point of failure as it will still be running in the other AZ, keeping the application up. Also, the cloud infrastructure provides automatic backup feature, so there is no such bottleneck like hard drive crash.

Assumptions to simplify cost analysis

- Comparing legacy system to only one cloud vendor, AWS
- Assuming setup and configuration costs are equivalent for both
- Not considering time used to implement the both systems
- Assuming no extra internet costs for both on-premise and cloud application

COST comparison of Legacy POS system and Cloud

Cloud computing is proving to be a great cost saving structure for many businesses. It provides its users with new age features like elasticity, availability, reliability, everything at lower costs in comparison to old systems. Cloud provides us with following cost structure:

- No upfront investments
- No costs for maintaining physical servers
- AWS offers cost-efficient options to choose from. Its three types of pricing offerings are: On-demand, Reserved and Spot Instances. In cloud, you only pay for what you use. To further reduce the costs if the use is consistent over the time, we can reserve instances in advance for 1-3 years. Also, there are spot instances where we bid on unused capacity and the prices are determined by the supply and demand of spots. In comparison to on-demand instances, these can help us save almost 80% of the price.

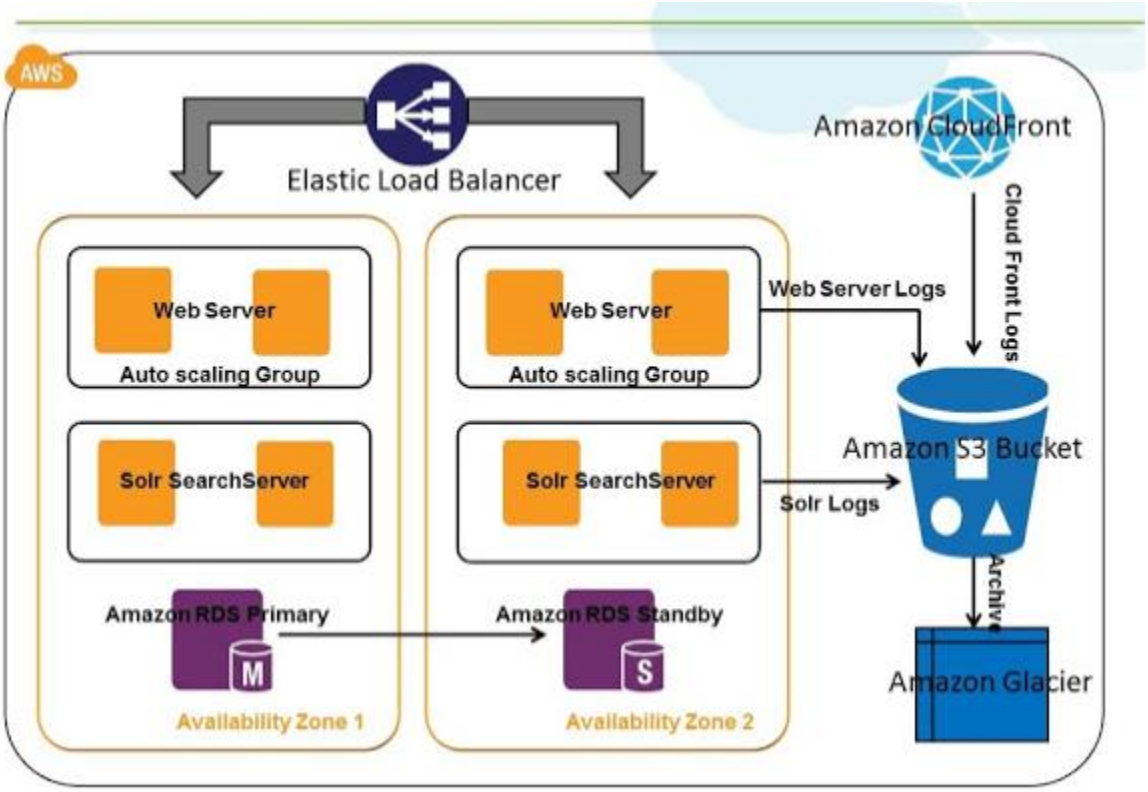
  Hence, from a cost stand point, Cloud applications are any day better than the legacy systems.

Risks/Cons involved in moving to the cloud

- **Required Internet Connection**: For accessing and retrieving the data from cloud, Starbucks will need an all time active internet connection. So, when the internet is down, that can be a problem when trying to access the data stored on cloud. Also, since the internet connection is a shared service, if a lot of people are using it at the same time, latency can increase.
- **Vendor lock-in**: When we start using services from a vendor, it becomes hard to move our data from it. It might be possible that when Starbucks start using AWS services it becomes hard to move to other vendor because of switching costs and this can prevent the company to use the benefits provided by cloud service completely.
- Lack of knowledge about the inside functioning of the network because everything will be managed by the cloud service provider.

Now the entire system is in the cloud, the system will function like this:

The AWS services like ELB, S3, RDS will play a vital role in backing up the application.



Hence, If we look at the overall picture, the benefits provided by Cloud services outweigh the risks involved. Also, when the costs are compared, migrating an application to cloud is any day a better solution. So, Migrating the application to cloud is a wise choice.

References: https://www.expertmarket.com/What-Point-Of-Sale-System-Does-Starbucks-Use
https://medium.com/aws-enterprise-collection/6-strategies-for-migrating-applications-to-the-cloud-eb4e85c412b4
https://www.qat.com/assets_articles/Legacy_Systems_Costing_You.pdf
http://searchcloudcomputing.techtarget.com/opinion/Clouds-are-more-secure-than-traditional-IT-systems-and-heres-why
https://www.linkedin.com/pulse/11-pros-cons-cloud-computing-everyone-should-know-umesh-singh/

Part 3: TECHNICAL SECTION

Summary: Created WordPress docker repository and used it to deploy in both AWS and Azure, but it is not stateless for both. The WordPress site is working for all the links provided. The links and credentials are provided at the beginning of each part.

Why Docker is better with real measurements?

Start and stop times

	Start Time	Stop Time
Docker Containers	<50ms	<50ms
VMs	30-45 seconds	5-10 seconds

- Containers churn 9 times faster than Virtual Machines. Container orchestration plays a vital role in container lifetimes, if the start and stop of containers is automated, this leads to high churn rate.
- Docker's abstraction provides us with isolation feature of the linux kernel to offer you a Virtual machine-like environment.
- **Memory:** Removes the overhead of running a guest OS, it requires consumed memory rather than provisioned memory.
- **CPU-overhead:** In comparison to a docker, kernel based virtual machine uses around 1.5% more CPU usage when idle**.**
- Docker provides a consistent environment with minimal performance overhead.
- Faster provisioning and portability helps us avoid vendor lock-in.

Which Cloud vendor would you use and why?

AWS is the biggest cloud hosting provider in the world, to support docker services it provides us with Elastic Container service(ECS). ECS shares a set of API calls to manage containers that we install across the EC2 instances. I definitely consider AWS a better docker service provider due to following features it offers:

- Consistent Environment: ECS consolidates the dependencies, libraries and all applications in one container, and ensures portability and provides us with a consistent environment.

- ECS allows you to track and maintain versions of a container, which helps in disaster recovery as when some unwanted changes happen, you can easily go back to previous version stored.
- ECS breaks the components of the application in multiple containers and makes sure there are no cross dependencies to allow user to upgrade any service independently.
- ECS is highly configurable and more secure as it allows you to launch the containers in your own VPC and use your own network ACLs.
- With AWS ECS, we can launch thousands of containers in seconds with no added complexity.

References: https://aws.amazon.com/ecs/

https://www.slideshare.net/Flux7Labs/performance-of-docker-vs-vms

https://medium.com/@betz.mark/comparing-amazon-elastic-container-service-and-google-kubernetes-1c63fbf19ccd

Technical project Execution Steps:

AWS credentials:
Username: tchaudhary@scu.edu
Password: P@ssw0rd!
EC2 Link: http://ec2-18-217-87-225.us-east-2.compute.amazonaws.com/

Azure credentials:
Username: tchaudhary@scu.edu
Password: cc_teacher
Azure WordPress link: http://tchaudhary123.westus.cloudapp.azure.com/

AWS:

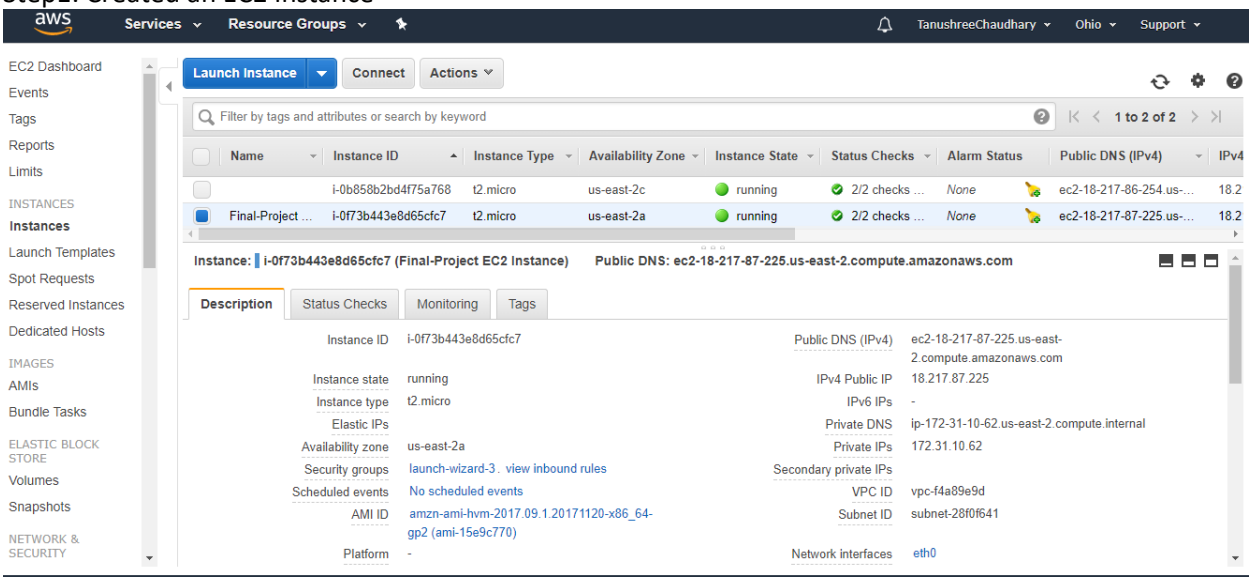
AWS credentials:

Username: tchaudhary@scu.edu

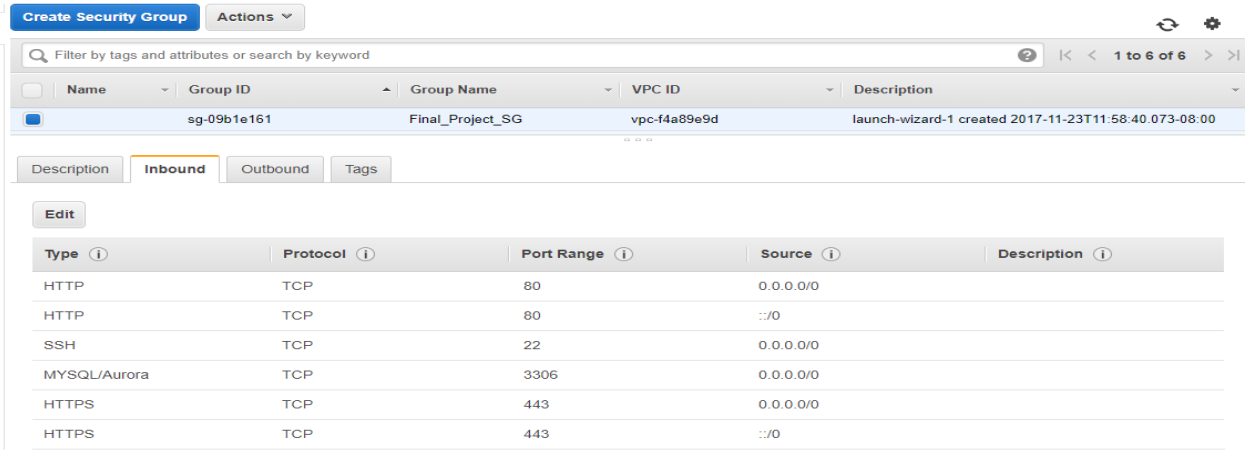
Password: P@ssw0rd!

EC2 Link: http://ec2-18-217-87-225.us-east-2.compute.amazonaws.com/

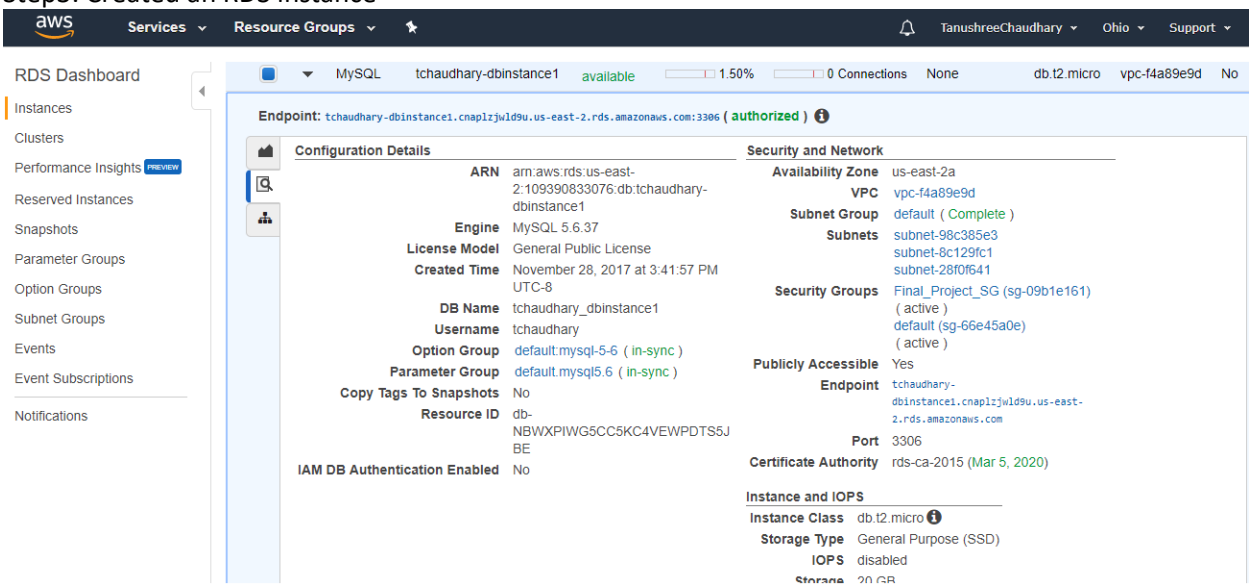
Step1: Created an EC2 instance



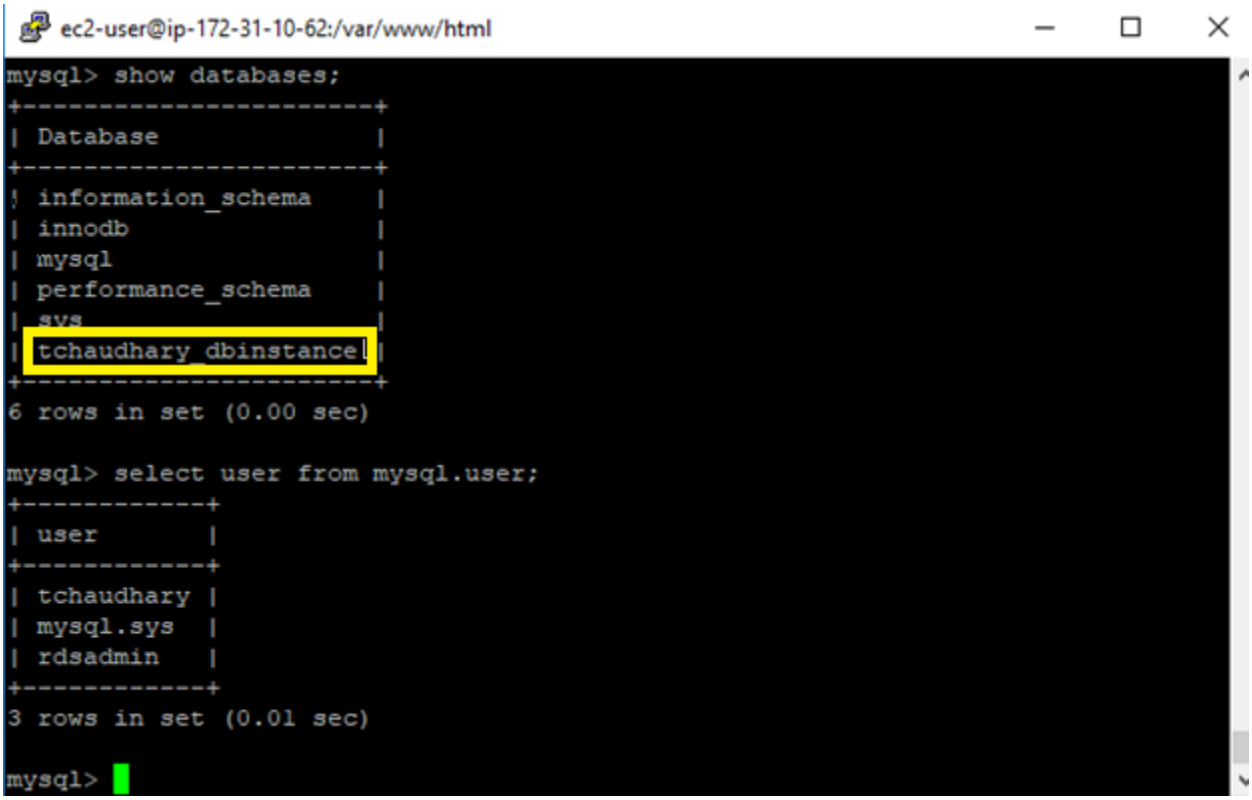
Security rules for the Instance



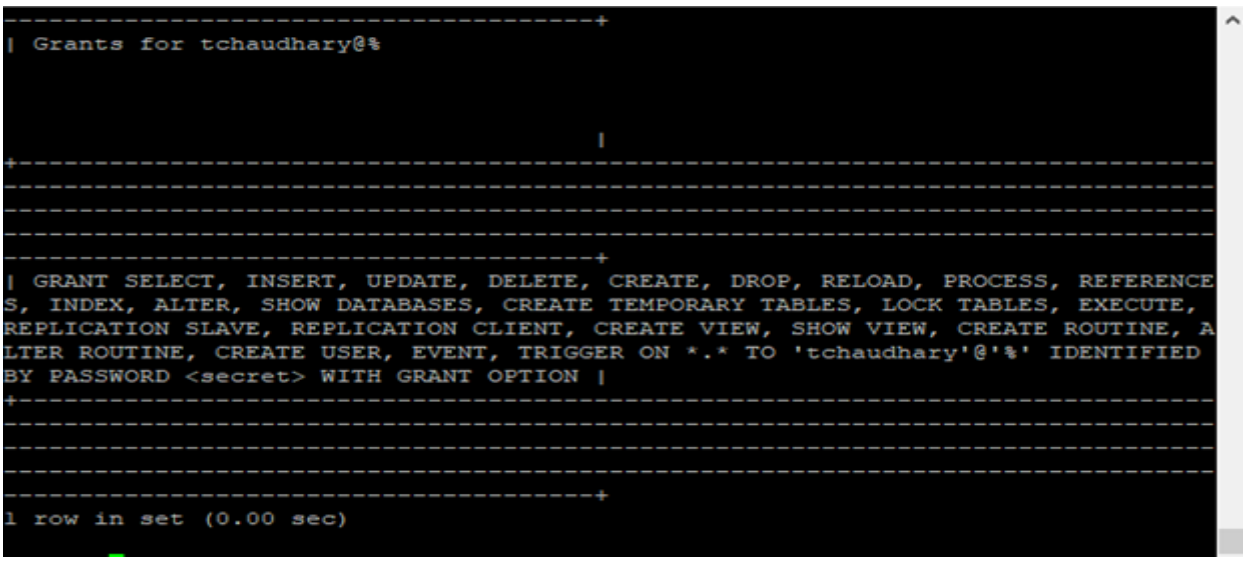
Step3: Created an RDS instance



Step4: Showing the database created



Step5: Showing the grants for the DB user



Step6: Showing the DB user

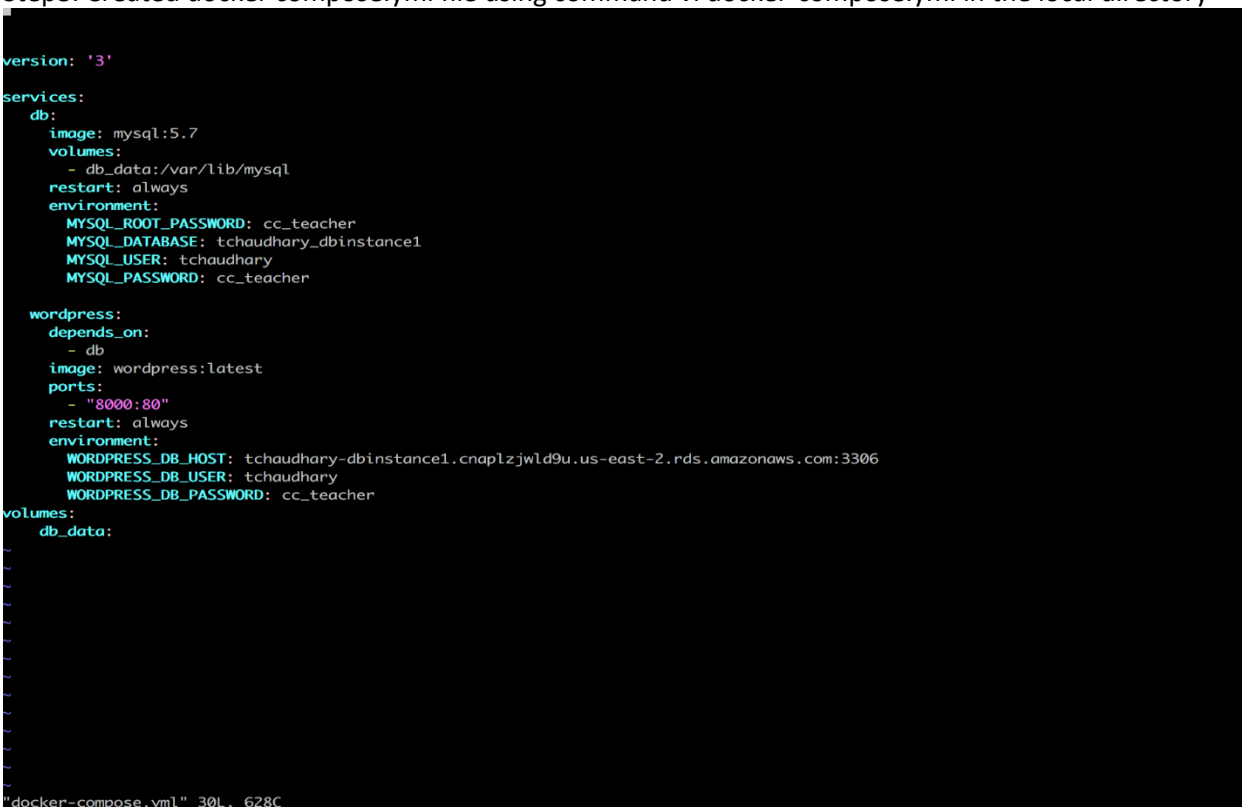
```
mysql> Select user from mysql.user;
+-------------+
| user        |
+-------------+
| tchaudhary  |
| mysql.sys   |
| rdsadmin    |
+-------------+
3 rows in set (0.00 sec)

mysql>
```

Step7: As docker was not working in my windows, I switched to mac for further steps. Edited the wp-config file in local directory.

wp-config-sample.php

```php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'tchaudhary_dbinstance1');

/** MySQL database username */
define('DB_USER', 'tchaudhary');

/** MySQL database password */
define('DB_PASSWORD', 'cc_teacher');

/** MySQL hostname */
define('DB_HOST', 'tchaudhary-dbinstance1.cnaplzjwld9u.us-east-2.rds.amazonaws.com');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

Step8: Created docker compose.yml file using command vi docker-compose.yml in the local directory



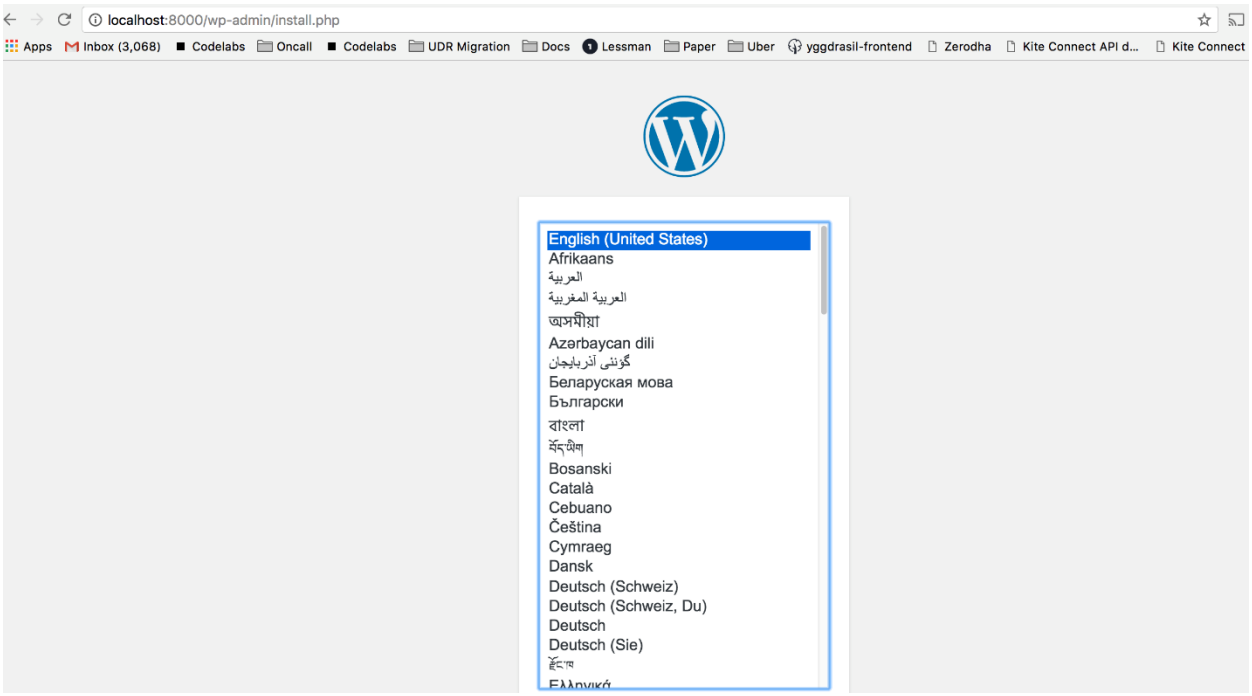
```
version: '3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: cc_teacher
       MYSQL_DATABASE: tchaudhary_dbinstance1
       MYSQL_USER: tchaudhary
       MYSQL_PASSWORD: cc_teacher

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: tchaudhary-dbinstance1.cnaplzjwld9u.us-east-2.rds.amazonaws.com:3306
       WORDPRESS_DB_USER: tchaudhary
       WORDPRESS_DB_PASSWORD: cc_teacher
volumes:
    db_data:
~
~
~
~
~
~
~
~
~
~
~
~
~
"docker-compose.yml" 30L, 628C
```

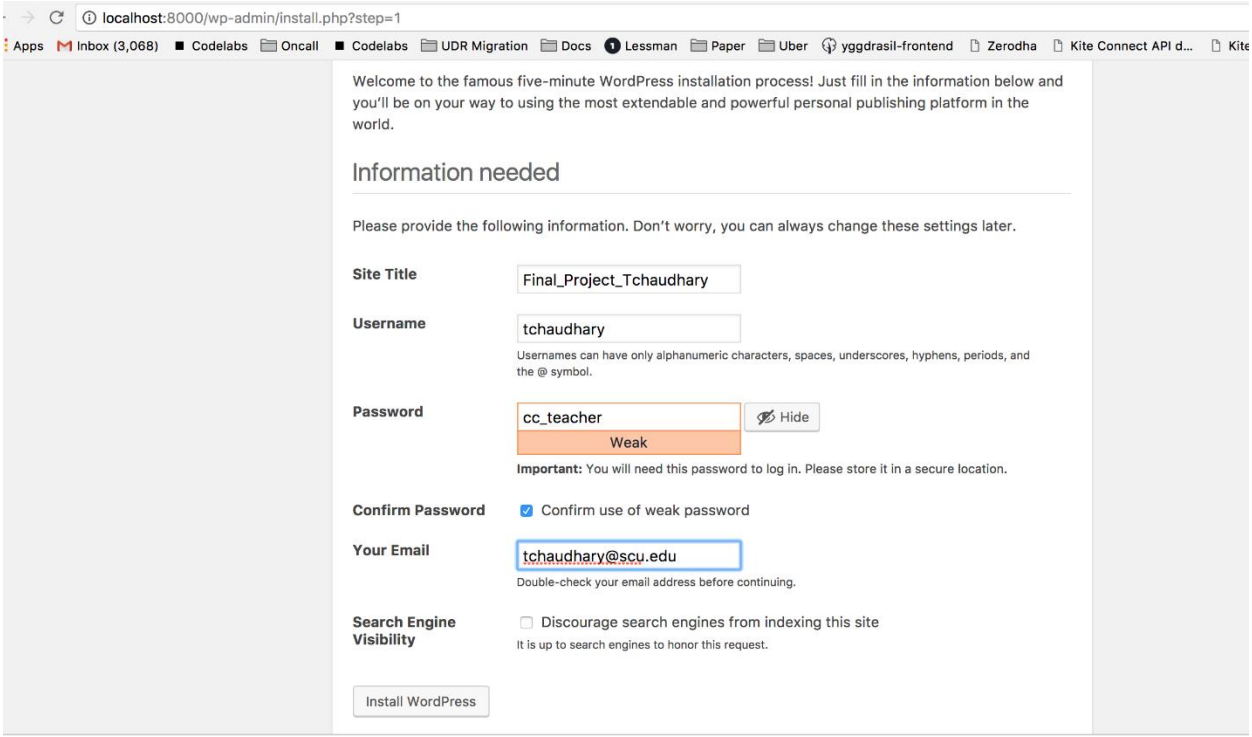
Step9: Run the docker-compose up -d command to push the files to the docker repository on cloud.

```
himank-C02PV3KHG8WM:applications himankchaudhary$ docker-compose up -d
WARNING: Dependency conflict: an older version of the 'docker-py' package may be polluting the namespace. If you're experiencing crashes, run
the following command to remedy the issue:
pip uninstall docker-py; pip uninstall docker; pip install docker
Creating network "applications_default" with the default driver
Creating volume "applications_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
85b1f47fba49: Pull complete
5671503d4f93: Pull complete
3b43b3b913cb: Pull complete
4fbb803665d0: Pull complete
05808866e6f9: Pull complete
1d8c65d48cfa: Pull complete
e189e187b2b5: Pull complete
02d3e6011ee8: Pull complete
d43b32d5ce04: Pull complete
2a809168ab45: Pull complete
Digest: sha256:1a2f9361228e9b10b4c77a651b460828514845dc7ac51735b919c2c4aec864b7
Status: Downloaded newer image for mysql:5.7
Pulling wordpress (wordpress: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
Creating applications_db_1 ...
Creating applications_db_1 ... done
Creating applications_wordpress_1 ...
```

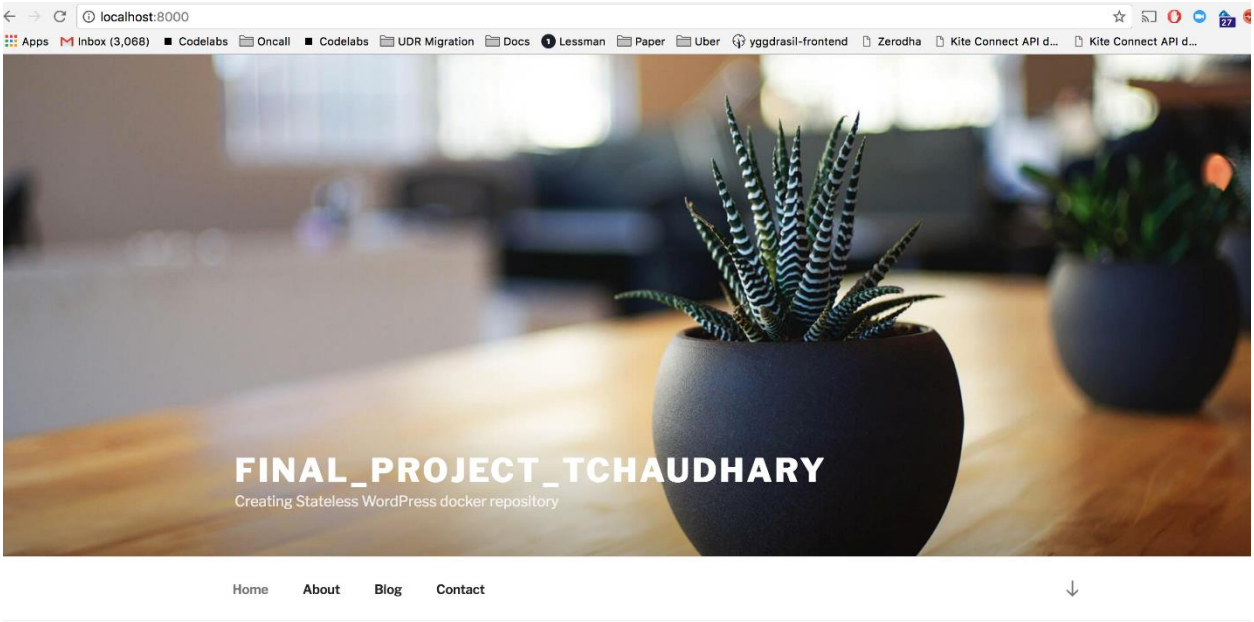
```
Status: Downloaded newer image for wordpress:latest
Creating applications_db_1 ...
Creating applications_db_1 ... done
Creating applications_wordpress_1 ...
Creating applications_wordpress_1 ... done
```

Step10: Install wordpress from the localhost:8000 once the docker-compose file is run successfully.


localhost:8000/wp-admin/install.php
Apps
Inbox (3,068)
Codelabs
Oncall
Codelabs
UDR Migration
Docs
Lessman
Paper
Uber
yggdrasil-frontend
Zerodha
Kite Connect API d...
Kite Connect
English (United States)
Afrikaans
العربية
العربية المغربية
অসমীয়া
Azərbaycan dili
گؤنئی آذربایجان
Беларуская мова
Български
বাংলা
བོད་ཡིག
Bosanski
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch (Schweiz)
Deutsch (Schweiz, Du)
Deutsch
Deutsch (Sie)


localhost:8000/wp-admin/install.php?step=1
Apps
Inbox (3,068)
Codelabs
Oncall
Codelabs
UDR Migration
Docs
Lessman
Paper
Uber
yggdrasil-frontend
Zerodha
Kite Connect API d...
Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.
Information needed
Please provide the following information. Don't worry, you can always change these settings later.
Site Title
Final_Project_Tchaudhary
Username
tchaudhary
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password
cc_teacher
Hide
Weak
Important: You will need this password to log in. Please store it in a secure location.
Confirm Password
Confirm use of weak password
Your Email
tchaudhary@scu.edu
Double-check your email address before continuing.
Search Engine Visibility
Discourage search engines from indexing this site
It is up to search engines to honor this request.
Install WordPress

WordPress Installed on localhost



Step11: Checked the docker images, it has wordpress and mysql with tags, latest and 5.7 respectively.

```
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
Creating applications_db_1 ...
Creating applications_db_1 ... done
Creating applications_wordpress_1 ...
Creating applications_wordpress_1 ... done
himank-C02PV3KHG8WM:applications himankchaudhary$ docker images
REPOSITORY                          TAG                                         IMAGE ID            CREATED             SIZE
wordpress                           latest                                      467f492bc127        8 days ago          413MB
127.0.0.1:15055/uber-usi/minimez    dca1-produ-0000000065                       631770289b0c        2 weeks ago         915MB
192.168.65.1:15055/uber-usi/minimez dca1-produ-0000000065                       631770289b0c        2 weeks ago         915MB
mysql                               5.7                                         5709795eeffa        3 weeks ago         408MB
dockerman                           63248acbc0185a32ee7edccbfc9acea9536f3b9c    f93c855afe1d        4 weeks ago         701MB
uber/ssh-agent-forward              latest                                      3c997745bd03        8 months ago        11MB
redis                               2.8                                         481995377a04        17 months ago       186MB
percona/percona-server              5.6.28                                      3ca2368807da        22 months ago       581MB
himank-C02PV3KHG8WM:applications himankchaudhary$
```

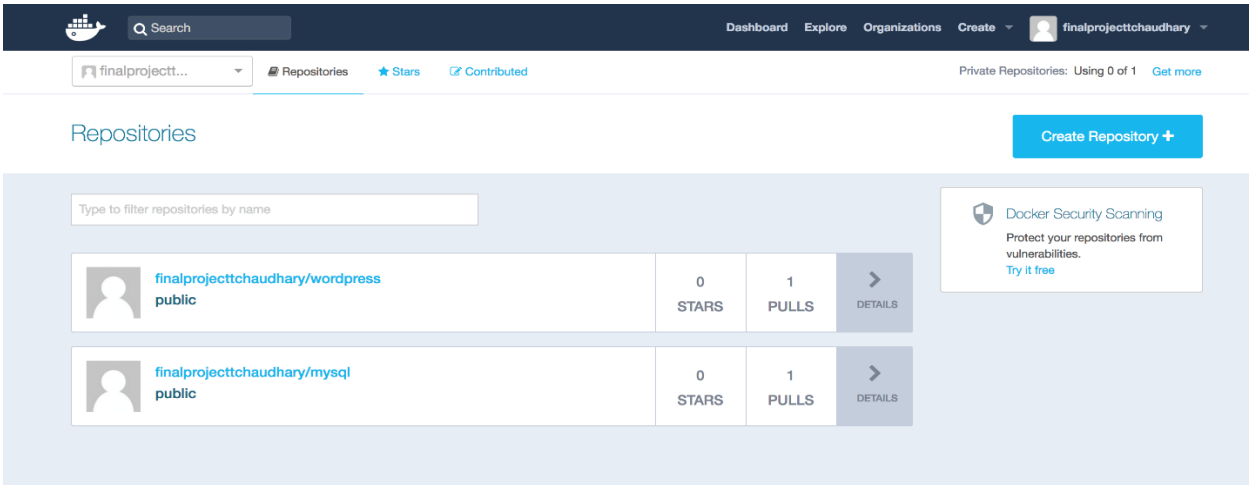
Step12: Tag the docker image wordpress to push to docker repository

```
himank-C02PV3KHG8WM:applications himankchaudhary$ docker tag wordpress:latest finalprojecttchaudhary/wordpress:latest
himank-C02PV3KHG8WM:applications himankchaudhary$ docker push finalprojecttchaudhary/wordpress:latest
The push refers to a repository [docker.io/finalprojecttchaudhary/wordpress]
d47b5306d1bf: Pushed
8911d4754681: Pushed
24605e7ca88b: Pushed
6594bf4ea5b9: Pushed
749e8aaa7dd4: Pushed
493137409f3e: Pushed
8933dc910eee: Pushed
61a961ab5d2b: Pushed
fa7f9311a060: Pushed
a9aa8861270e: Pushed
dcdbe9fe2ca1: Pushed
2f6273a5f133: Pushed
4c0354ed71f4: Pushed
2c3aa4e96952: Pushed
5cd2e0cfe892: Pushed
c3d26400d3ff: Pushed
37412c153883: Pushed
c01c63c6823d: Pushed
latest: digest: sha256:5b3b36db3c19d5b8c6ded6facec4daac57fe2ea1879351a2e65ac8919cea37ce size: 4078
```

Step13: Tag the docker image mysql to push to docker repository

```
himank-C02PV3KHG8WM:applications himankchaudhary$ docker tag mysql:5.7 finalprojecttchaudhary/mysql:5.7
himank-C02PV3KHG8WM:applications himankchaudhary$ docker push finalprojecttchaudhary/mysql:5.7
The push refers to a repository [docker.io/finalprojecttchaudhary/mysql]
07a2fc7ae711: Mounted from library/mysql
6d4479921db2: Mounted from library/mysql
e3608fd655a7: Mounted from library/mysql
9cb944e05700: Mounted from library/mysql
5736c5b60420: Mounted from library/mysql
7bf08445116f: Mounted from library/mysql
9545e4b14d5a: Mounted from library/mysql
b6786fdea2fc: Mounted from library/mysql
10d8a8f4f236: Mounted from library/mysql
c01c63c6823d: Mounted from finalprojecttchaudhary/wordpress
5.7: digest: sha256:a0423a7d021b7a7775f1d2db1014bd15fde029f538c1f8d97c9832aa4a25209f size: 2410
himank-C02PV3KHG8WM:applications himankchaudhary$
```

Step14: Docker Repository with pushed images



Step15: From docker repository, pulling image into ec2 instance

```
Bye
[ec2-user@ip-172-31-34-138 ~]$ sudo yum update -y
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main                                                                | 2.1 kB  00:00:00
amzn-updates                                                             | 2.5 kB  00:00:00
No packages marked for update
[ec2-user@ip-172-31-34-138 ~]$ sudo yum install -y docker
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:17.06.2ce-1.93.amzn1 will be installed
--> Processing Dependency: xfsprogs for package: docker-17.06.2ce-1.93.amzn1.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package: docker-17.06.2ce-1.93.amzn1.x86_64
--> Processing Dependency: libltdl.so.7()(64bit) for package: docker-17.06.2ce-1.93.amzn1.x86_64
--> Running transaction check
---> Package libseccomp.x86_64 0:2.3.1-2.4.amzn1 will be installed
---> Package libtool-ltdl.x86_64 0:2.4.2-20.4.8.5.32.amzn1 will be installed
---> Package xfsprogs.x86_64 0:4.5.0-9.21.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch                Version                  Repository                Size
================================================================================
Installing:
 docker                 x86_64              17.06.2ce-1.93.amzn1     amzn-updates              25 M
Installing for dependencies:
 libseccomp             x86_64              2.3.1-2.4.amzn1          amzn-main                 79 k
 libtool-ltdl           x86_64              2.4.2-20.4.8.5.32.amzn1  amzn-main                 51 k
 xfsprogs               x86_64              4.5.0-9.21.amzn1         amzn-main                1.7 M

Transaction Summary
================================================================================
Install  1 Package (+3 Dependent packages)

Total download size: 27 M
Installed size: 80 M
Downloading packages:
(1/4): libtool-ltdl-2.4.2-20.4.8.5.32.amzn1.x86_64.rpm                   |  51 kB  00:00:00
(2/4): libseccomp-2.3.1-2.4.amzn1.x86_64.rpm                             |  79 kB  00:00:00
(3/4): xfsprogs-4.5.0-9.21.amzn1.x86_64.rpm                              | 1.7 MB  00:00:00
```

```
================================================================================
 Package                Arch                Version                  Repository                Size
================================================================================
Installing:
 docker                 x86_64              17.06.2ce-1.93.amzn1     amzn-updates              25 M
Installing for dependencies:
 libseccomp             x86_64              2.3.1-2.4.amzn1          amzn-main                 79 k
 libtool-ltdl           x86_64              2.4.2-20.4.8.5.32.amzn1  amzn-main                 51 k
 xfsprogs               x86_64              4.5.0-9.21.amzn1         amzn-main                1.7 M

Transaction Summary
================================================================================
Install  1 Package (+3 Dependent packages)

Total download size: 27 M
Installed size: 80 M
Downloading packages:
(1/4): libtool-ltdl-2.4.2-20.4.8.5.32.amzn1.x86_64.rpm                    |  51 kB  00:00:00
(2/4): libseccomp-2.3.1-2.4.amzn1.x86_64.rpm                              |  79 kB  00:00:00
(3/4): xfsprogs-4.5.0-9.21.amzn1.x86_64.rpm                               | 1.7 MB  00:00:00
(4/4): docker-17.06.2ce-1.93.amzn1.x86_64.rpm                             |  25 MB  00:00:03
--------------------------------------------------------------------------------
Total                                                  8.0 MB/s |  27 MB  00:00:03
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libseccomp-2.3.1-2.4.amzn1.x86_64                                1/4
  Installing : libtool-ltdl-2.4.2-20.4.8.5.32.amzn1.x86_64                      2/4
  Installing : xfsprogs-4.5.0-9.21.amzn1.x86_64                                 3/4
  Installing : docker-17.06.2ce-1.93.amzn1.x86_64                               4/4
  Verifying  : docker-17.06.2ce-1.93.amzn1.x86_64                               1/4
  Verifying  : xfsprogs-4.5.0-9.21.amzn1.x86_64                                 2/4
  Verifying  : libtool-ltdl-2.4.2-20.4.8.5.32.amzn1.x86_64                      3/4
  Verifying  : libseccomp-2.3.1-2.4.amzn1.x86_64                                4/4

Installed:
  docker.x86_64 0:17.06.2ce-1.93.amzn1

Dependency Installed:
  libseccomp.x86_64 0:2.3.1-2.4.amzn1        libtool-ltdl.x86_64 0:2.4.2-20.4.8.5.32.amzn1        xfsprogs.x86_64 0:4.5.0-9.21.amzn1

Complete!
[ec2-user@ip-172-31-34-138 ~]$
```

Step16: Showing docker Info

```
[ec2-user@ip-172-31-34-138 ~]$ docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 17.06.2-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 6e23458c129b551d5c9871e5174f6b1b7f6d1170
runc version: 810190ceaa507aa2727d7ae6f4790c76ec150bd2
init version: 949e6fa
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.9.62-21.56.amzn1.x86_64
Operating System: Amazon Linux AMI 2017.09
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 993.4MiB
Name: ip-172-31-34-138
ID: TDIB:ESFK:PFHK:4WDK:M5NH:PN3E:AXCZ:RUF3:2CZZ:XNSH:TP44:Y3VS
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

[ec2-user@ip-172-31-34-138 ~]$
```

Step17: Wordpress has been successfully copied to /var/www/html

```
[ec2-user@ip-172-31-34-138 ~]$ docker run -it --rm -p 3000:8080 finalprojecttchaudhary/wordpress
Unable to find image 'finalprojecttchaudhary/wordpress:latest' locally
latest: Pulling from finalprojecttchaudhary/wordpress
85b1f47fba49: Pull complete
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: sha256:5b3b36db3c19d5b8c6ded6facec4daac57fe2ea1879351a2e65ac8919cea37ce
Status: Downloaded newer image for finalprojecttchaudhary/wordpress:latest
WordPress not found in /var/www/html - copying now...
Complete! WordPress has been successfully copied to /var/www/html
```

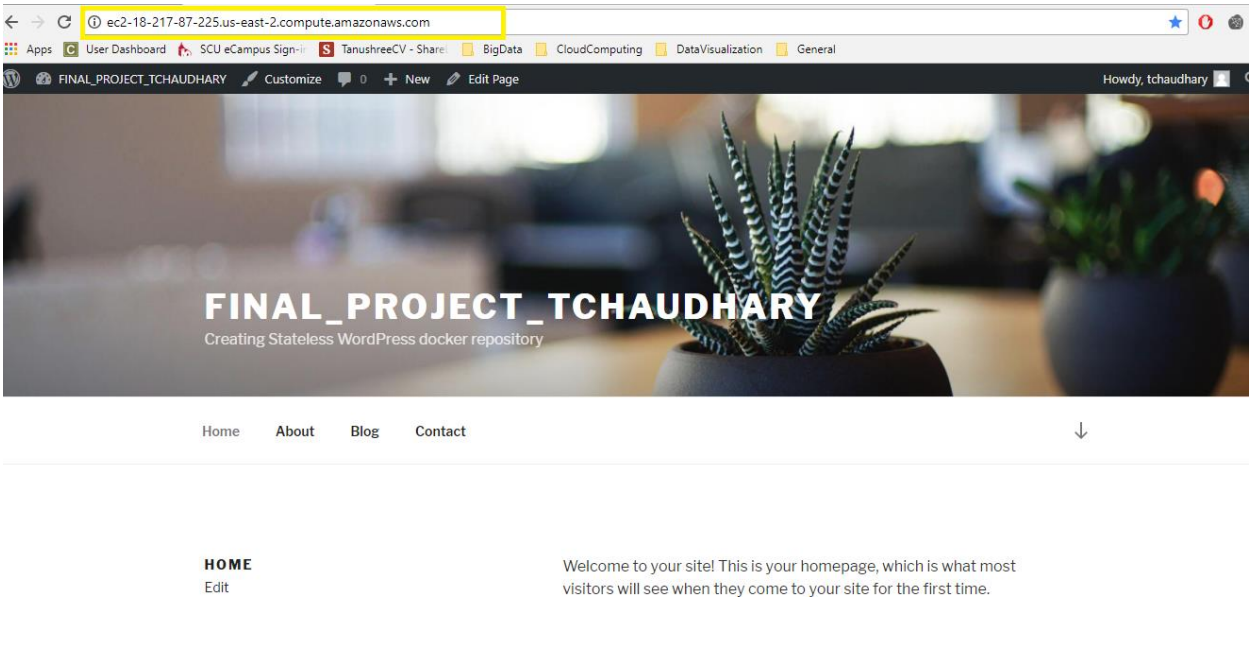
Step18: Docker run for mysql

```
[ec2-user@ip-172-31-34-138 ~]$ docker run -p 80:80 finalprojecttchaudhary/mysql:5.7
Unable to find image 'finalprojecttchaudhary/mysql:5.7' locally
5.7: Pulling from finalprojecttchaudhary/mysql
85b1f47fba49: Already exists
5671503d4f93: Pull complete
3b43b3b913cb: Pull complete
4fbb803665d0: Pull complete
05808866e6f9: Pull complete
1d8c65d48cfa: Pull complete
e189e187b2b5: Pull complete
02d3e6011ee8: Pull complete
d43b32d5ce04: Pull complete
2a809168ab45: Pull complete
Digest: sha256:a0423a7d021b7a7775f1d2db1014bd15fde029f538c1f8d97c9832aa4a25209f
Status: Downloaded newer image for finalprojecttchaudhary/mysql:5.7
docker: Error response from daemon: driver failed programming external connectivity on endpoint sad_allen (8c686060247bcb865977138add00510c92b0
26a4de377ee2250906d435e6b792): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: address already in use.
```

Step19: Displaying docker repository

```
[ec2-user@ip-172-31-34-138 ~]$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
finalprojecttchaudhary/wordpress    latest              467f492bc127        8 days ago          413MB
finalprojecttchaudhary/mysql        5.7                 5709795eeffa        3 weeks ago         408MB
[ec2-user@ip-172-31-34-138 ~]$
```

Step20: EC2 link displaying the WordPress
http://ec2-18-217-87-225.us-east-2.compute.amazonaws.com/



Azure:

Azure credentials:
Username: tchaudhary@scu.edu
Password: cc_teacher
Azure WordPress link: http://tchaudhary123.westus.cloudapp.azure.com/

Step1: Downloaded Azure CLI and logged into azure account through CLI. Create docker host with Azure CLI. Created a resource group name, RG in West US location.

```
himank-C02PV3KHG8WM:~ himankchaudhary$ az login
To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code B7GSZSEFC to authenticate.
[
  {
    "cloudName": "AzureCloud",
    "id": "1f688910-4854-4b80-aa3a-d420d95feb5b",
    "isDefault": true,
    "name": "Free Trial",
    "state": "Enabled",
    "tenantId": "f1f02bc1-60c0-48e0-82bc-069150bd1e36",
    "user": {
      "name": "tchaudhary@scu.edu",
      "type": "user"
    }
  }
]
himank-C02PV3KHG8WM:~ himankchaudhary$ az group create --name RG --location westus
{
  "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG",
  "location": "westus",
  "managedBy": null,
  "name": "RG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

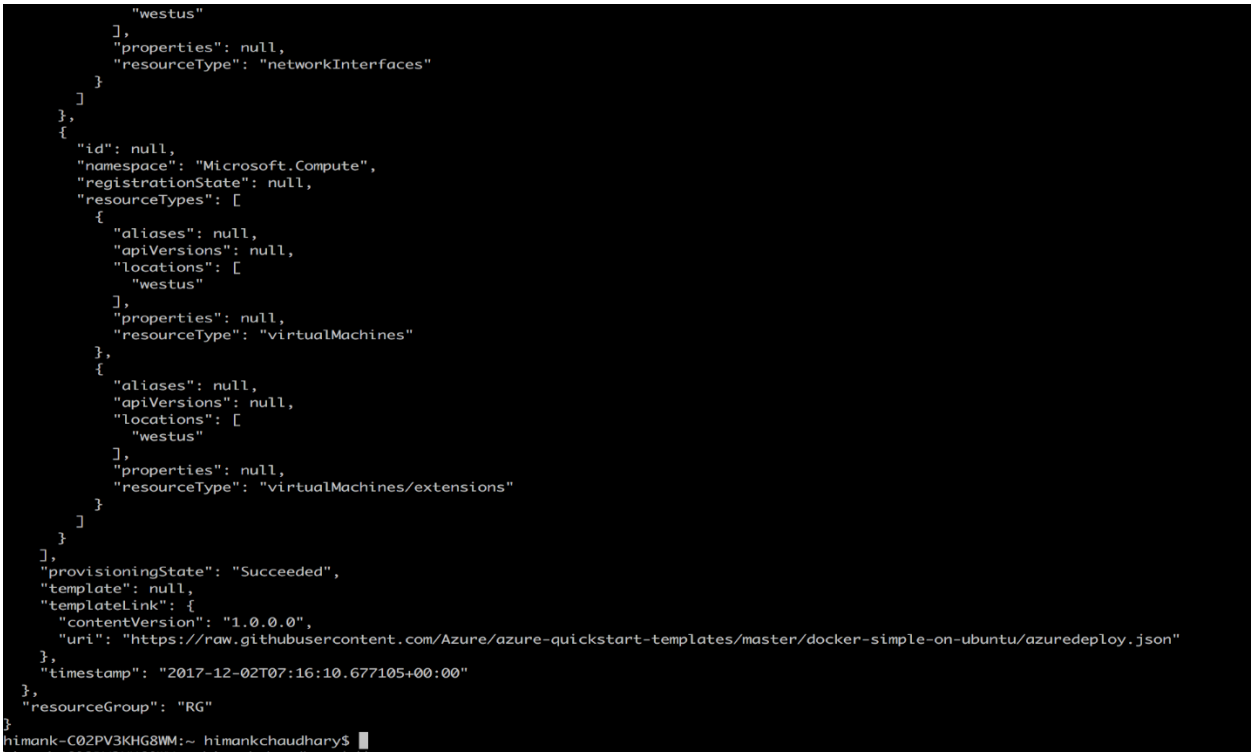
Step2: Created a deployment with storage account, admin username, password and dns name parameters

```
himank-C02PV3KHG8WM:~ himankchaudhary$ az group deployment create --resource-group RG \
>   --parameters '{"newStorageAccountName": {"value": "finalprojectdocker"},
>     "adminUsername": {"value": "tchaudhary"},
>     "adminPassword": {"value": "P@ssw0rd!"},
>     "dnsNameForPublicIP": {"value": "tchaudhary123"}}' \
>   --template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/docker-simple-on-ubuntu/azuredeploy.json
{
  "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Resources/deployments/azuredeploy",
  "name": "azuredeploy",
  "properties": {
    "correlationId": "ec8c1302-c99c-44ce-9853-8c071a80f6b8",
    "debugSetting": null,
    "dependencies": [
      {
        "dependsOn": [
          {
            "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Network/publicIPAddresses/myPublicIPD",
            "resourceGroup": "RG",
            "resourceName": "myPublicIPD",
            "resourceType": "Microsoft.Network/publicIPAddresses"
          },
          {
            "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Network/virtualNetworks/MyVNETD",
            "resourceGroup": "RG",
            "resourceName": "MyVNETD",
            "resourceType": "Microsoft.Network/virtualNetworks"
          }
        ],
        "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Network/networkInterfaces/myVMNicD",
        "resourceGroup": "RG",
        "resourceName": "myVMNicD",
        "resourceType": "Microsoft.Network/networkInterfaces"
      },
      {
        "dependsOn": [
          {
            "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Storage/storageAccounts/finalprojectdocker",
            "resourceGroup": "RG",
            "resourceName": "finalprojectdocker",
            "resourceType": "Microsoft.Storage/storageAccounts"
          },
          {
            "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Network/networkInterfaces/myVMNicD",
            "resourceGroup": "RG",
```

```
          {
            "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Network/networkInterfaces/myVMNicD",
            "resourceGroup": "RG",
            "resourceName": "myVMNicD",
            "resourceType": "Microsoft.Network/networkInterfaces"
          }
        ],
        "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Compute/virtualMachines/MyDockerVM",
        "resourceGroup": "RG",
        "resourceName": "MyDockerVM",
        "resourceType": "Microsoft.Compute/virtualMachines"
      },
      {
        "dependsOn": [
          {
            "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Compute/virtualMachines/MyDockerVM",
            "resourceGroup": "RG",
            "resourceName": "MyDockerVM",
            "resourceType": "Microsoft.Compute/virtualMachines"
          }
        ],
        "id": "/subscriptions/1f688910-4854-4b80-aa3a-d420d95feb5b/resourceGroups/RG/providers/Microsoft.Compute/virtualMachines/MyDockerVM/extensions/DockerExtension
",
        "resourceGroup": "RG",
        "resourceName": "MyDockerVM/DockerExtension",
        "resourceType": "Microsoft.Compute/virtualMachines/extensions"
      }
    ],
    "mode": "Incremental",
    "outputs": null,
    "parameters": {
      "adminPassword": {
        "type": "SecureString"
      },
      "adminUsername": {
        "type": "String",
        "value": "tchaudhary"
      },
      "dnsNameForPublicIP": {
        "type": "String",
        "value": "tchaudhary123"
      },
      "newStorageAccountName": {
        "type": "String",
```

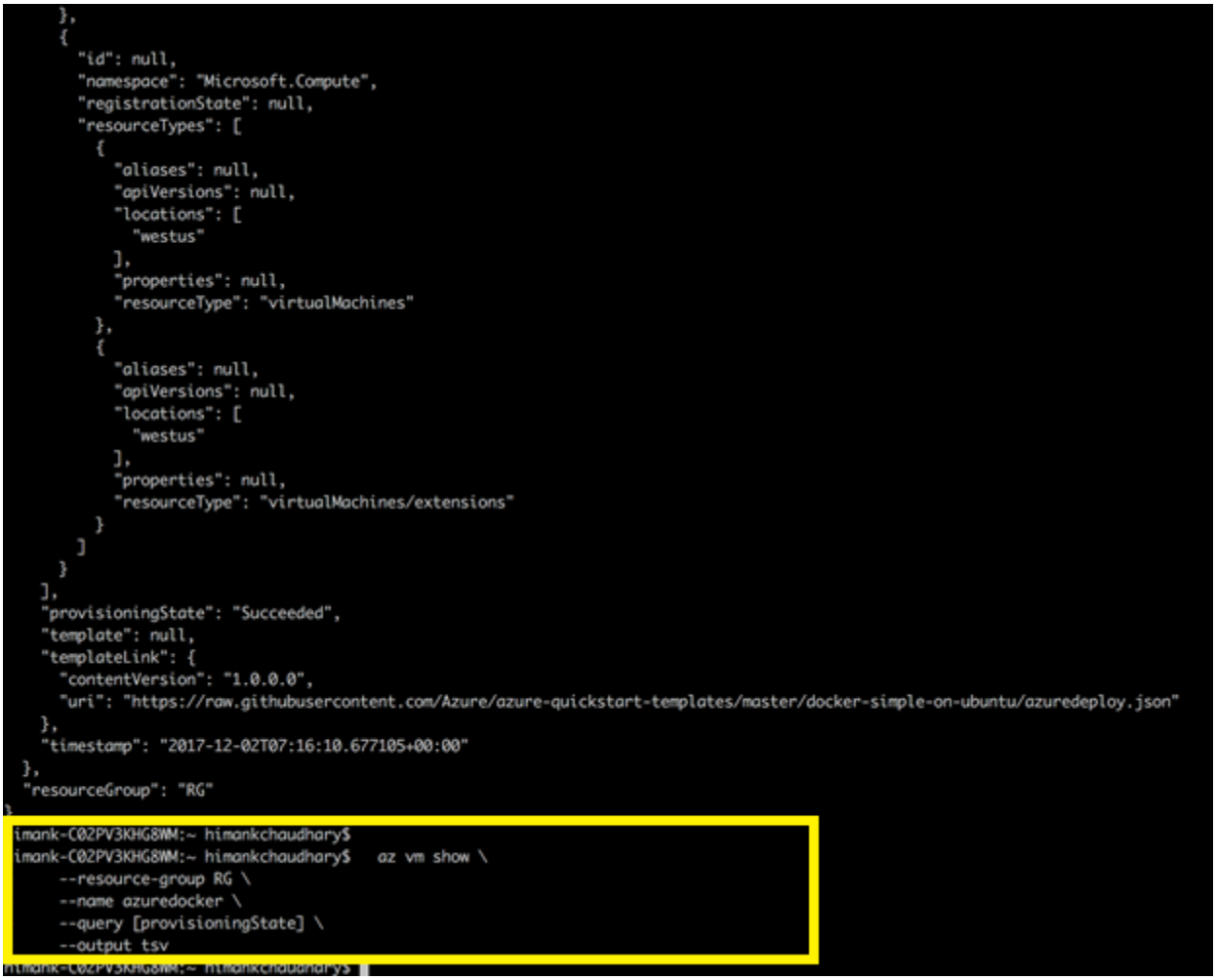
Details about the deployment created.


```
    "newStorageAccountName": {
      "type": "String",
      "value": "finalprojectdocker"
    },
    "ubuntuOSVersion": {
      "type": "String",
      "value": "14.04.4-LTS"
    },
    "vmSize": {
      "type": "String",
      "value": "Standard_F1"
    }
  },
  "parametersLink": null,
  "providers": [
    {
      "id": null,
      "namespace": "Microsoft.Storage",
      "registrationState": null,
      "resourceTypes": [
        {
          "aliases": null,
          "apiVersions": null,
          "locations": [
            "westus"
          ],
          "properties": null,
          "resourceType": "storageAccounts"
        }
      ]
    },
    {
      "id": null,
      "namespace": "Microsoft.Network",
      "registrationState": null,
      "resourceTypes": [
        {
          "aliases": null,
          "apiVersions": null,
          "locations": [
            "westus"
          ],
          "properties": null,
          "resourceType": "publicIPAddresses"
```



Done with creating the deployment.

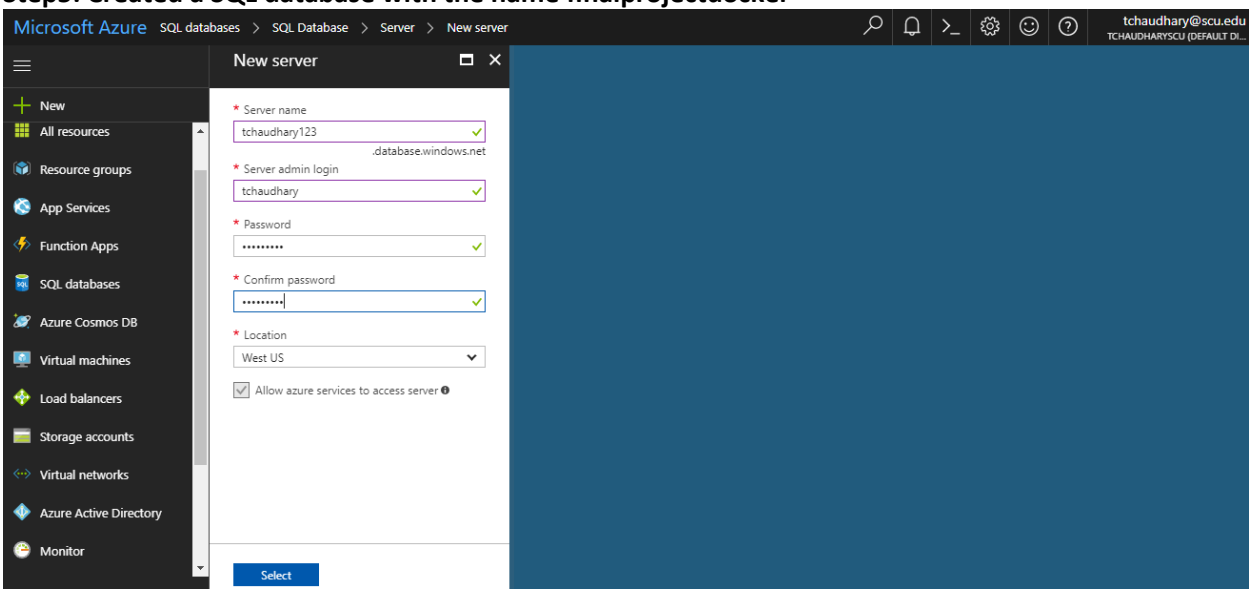
Step3: Verifying that compose is installed

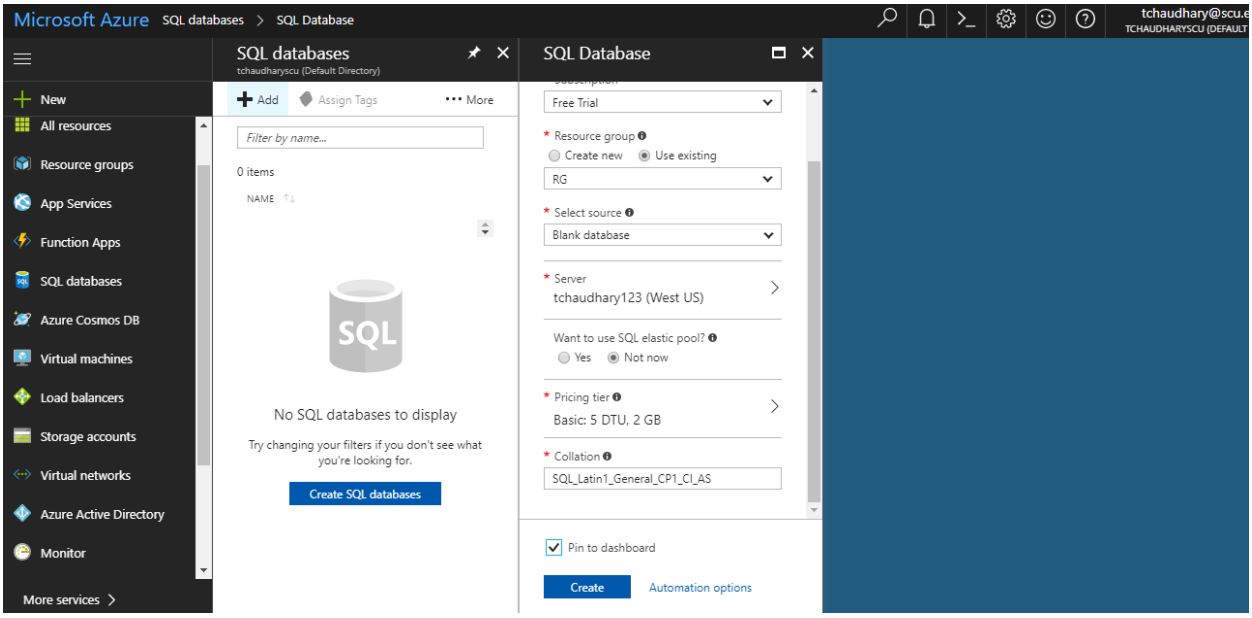


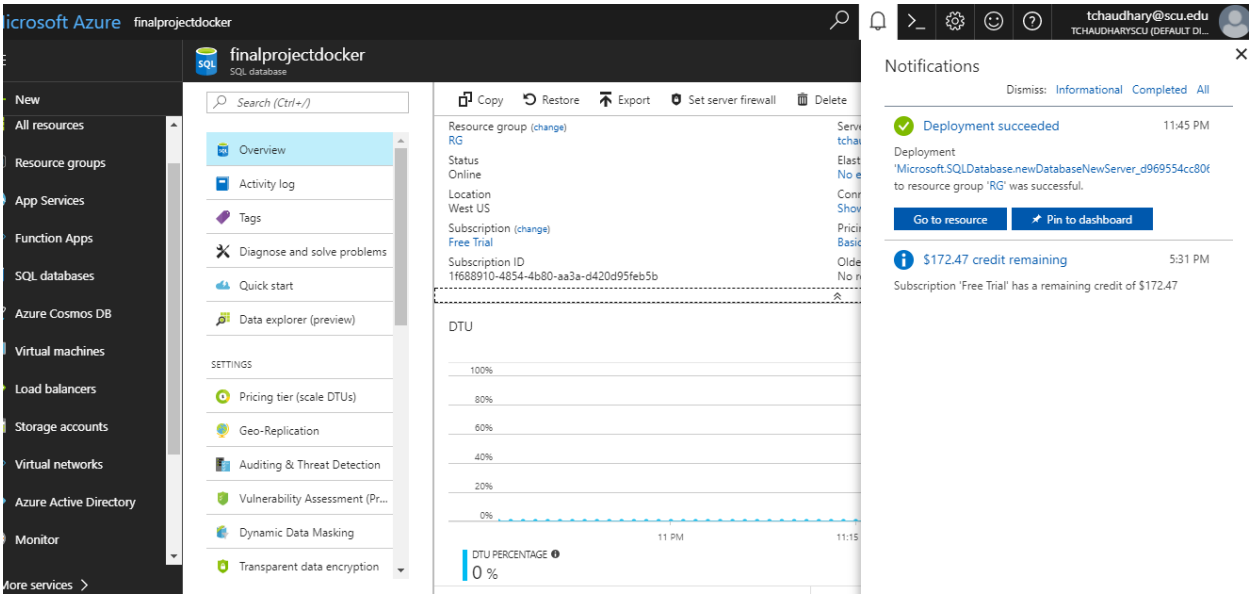
Step4: SSH into the new docker host by providing my own dns name.
DNS name: tchaudhary@tchaudhary123.westus.cloudapp.azure.com

```
himank-C02PV3KHG8WM:~ himankchaudhary$  ssh tchaudhary@tchaudhary123.westus.cloudapp.azure.com
tchaudhary@tchaudhary123.westus.cloudapp.azure.com's password:
Permission denied, please try again.
tchaudhary@tchaudhary123.westus.cloudapp.azure.com's password:
Permission denied, please try again.
tchaudhary@tchaudhary123.westus.cloudapp.azure.com's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.19.0-65-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sat Dec  2 07:13:22 UTC 2017

  System load: 0.23              Memory usage: 5%   Processes:       83
  Usage of /:  39.6% of 1.94GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

tchaudhary@MyDockerVM:~$
```

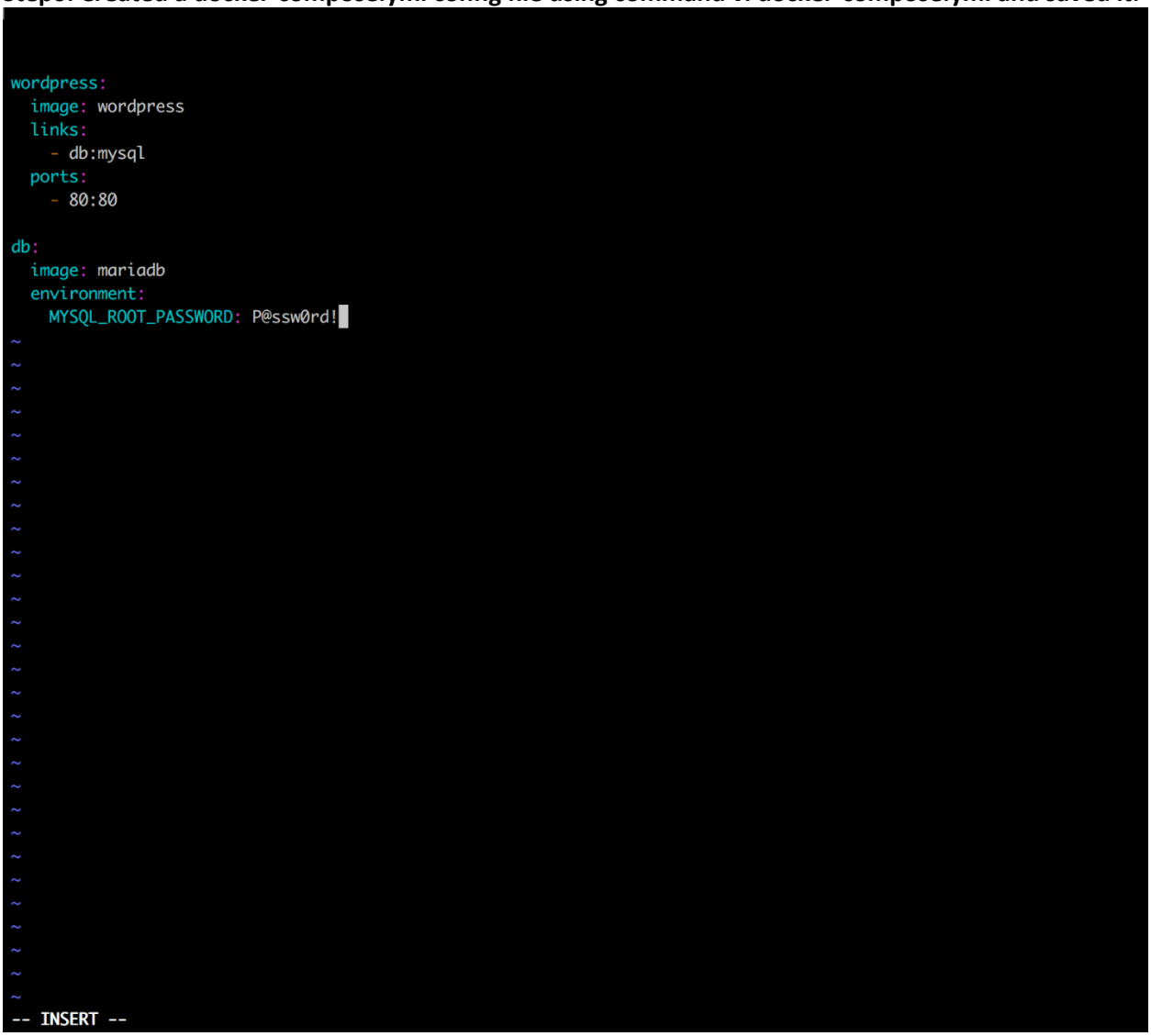
Step5: Created a SQL database with the name finalprojectdocker




Microsoft Azure
SQL databases > SQL Database
tchaudhary@scu.e
TCHAUDHARYSCU (DEFAULT
SQL databases
tchaudharyscu (Default Directory)
New
All resources
Resource groups
App Services
Function Apps
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Monitor
More services
Add
Assign Tags
More
Filter by name...
0 items
NAME
SQL
No SQL databases to display
Try changing your filters if you don't see what you're looking for.
Create SQL databases
SQL Database
Free Trial
* Resource group
Create new
Use existing
RG
* Select source
Blank database
* Server
tchaudhary123 (West US)
Want to use SQL elastic pool?
Yes
Not now
* Pricing tier
Basic: 5 DTU, 2 GB
* Collation
SQL_Latin1_General_CP1_CI_AS
Pin to dashboard
Create
Automation options


icrosoft Azure
finalprojectdocker
tchaudhary@scu.edu
TCHAUDHARYSCU (DEFAULT DI...
finalprojectdocker
SQL database
New
All resources
Resource groups
App Services
Function Apps
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Monitor
More services
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Data explorer (preview)
SETTINGS
Pricing tier (scale DTUs)
Geo-Replication
Auditing & Threat Detection
Vulnerability Assessment (Pr...
Dynamic Data Masking
Transparent data encryption
Copy
Restore
Export
Set server firewall
Delete
Resource group (change)
RG
Status
Online
Location
West US
Subscription (change)
Free Trial
Subscription ID
1f688910-4854-4b80-aa3a-d420d95feb5b
DTU
100%
80%
60%
40%
20%
0%
11 PM
DTU PERCENTAGE
0 %
Notifications
Dismiss: Informational Completed All
Deployment succeeded
11:45 PM
Deployment
'Microsoft.SQLDatabase.newDatabaseNewServer_d969554cc80
to resource group 'RG' was successful.
Go to resource
Pin to dashboard
$172.47 credit remaining
5:31 PM
Subscription 'Free Trial' has a remaining credit of $172.47

Step6: Created a docker-compose.yml config file using command vi docker-compose.yml and saved it.



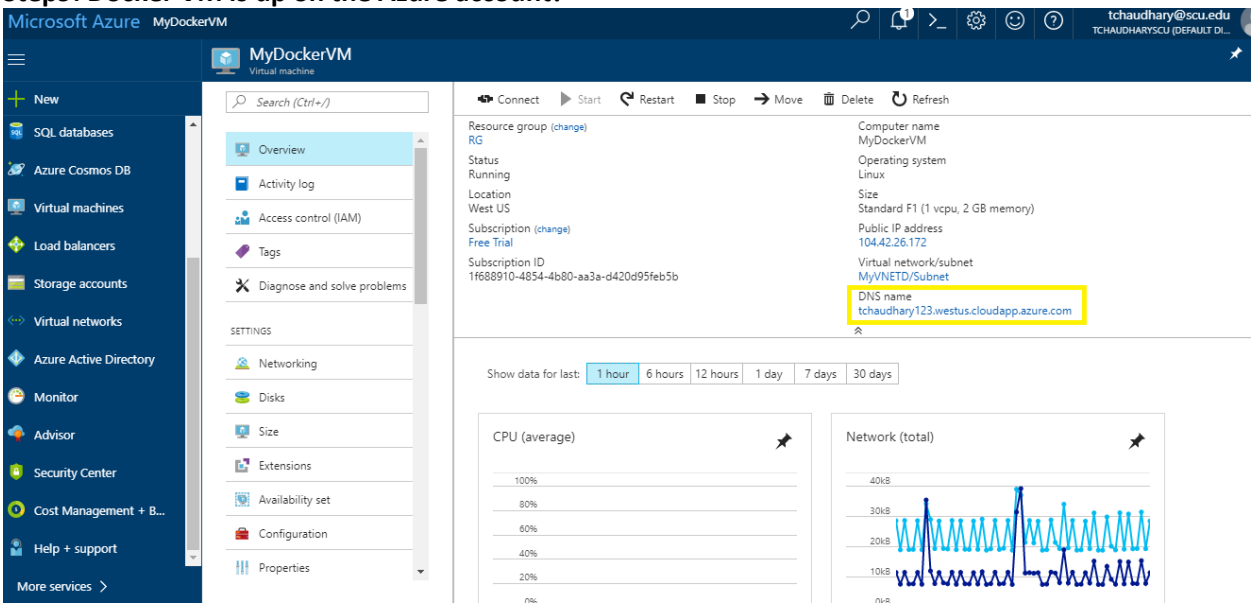
Step7: Starting the containers with the compose using docker-compose up -d

```
tchaudhary@MyDockerVM:~$ docker-compose --version
docker-compose version 1.6.2, build 4d72027
tchaudhary@MyDockerVM:~$ vi docker-compose.yml
tchaudhary@MyDockerVM:~$ docker-compose up -d
Pulling db (mariadb:latest)...
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
abe17ed7eabe: Pull complete
a5488a3ed5dd: Pull complete
6263a21bc821: Pull complete
bb5bd77424aa: Pull complete
Digest: sha256:d840bd45e8b1da7e4a77c0610a0f9a5b1f15d960268cf089f89539c658e15cb6
Status: Downloaded newer image for mariadb:latest
Creating tchaudhary_db_1
Pulling wordpress (wordpress:latest)...
latest: Pulling from library/wordpress
85b1f47fba49: Already exists
c266283065ed: Pull complete
ec99b99d75b1: Pull complete
1acd20a385df: Pull complete
82d806755bbb: Pull complete
0a68419027d0: Pull complete
1d54cf8a3cef: Pull complete
7f625202d40e: Pull complete
8d44d50012c4: Pull complete
bde2597589da: Pull complete
ffc7ed272edc: Pull complete
0a03861917b8: Pull complete
ac9c6b3bf298: Pull complete
0728fc0e0c7f: Pull complete
30bb9a0b7851: Pull complete
6c6d25d2ce66: Pull complete
b6b18f55e533: Pull complete
49c5f1d693cd: Pull complete
Digest: sha256:8fd3cad0d1a9291db828ea74a7aee4cc01ff94b5ee17df493279e4d673cab56f
Status: Downloaded newer image for wordpress:latest
Creating tchaudhary_wordpress_1
tchaudhary@MyDockerVM:~$
```

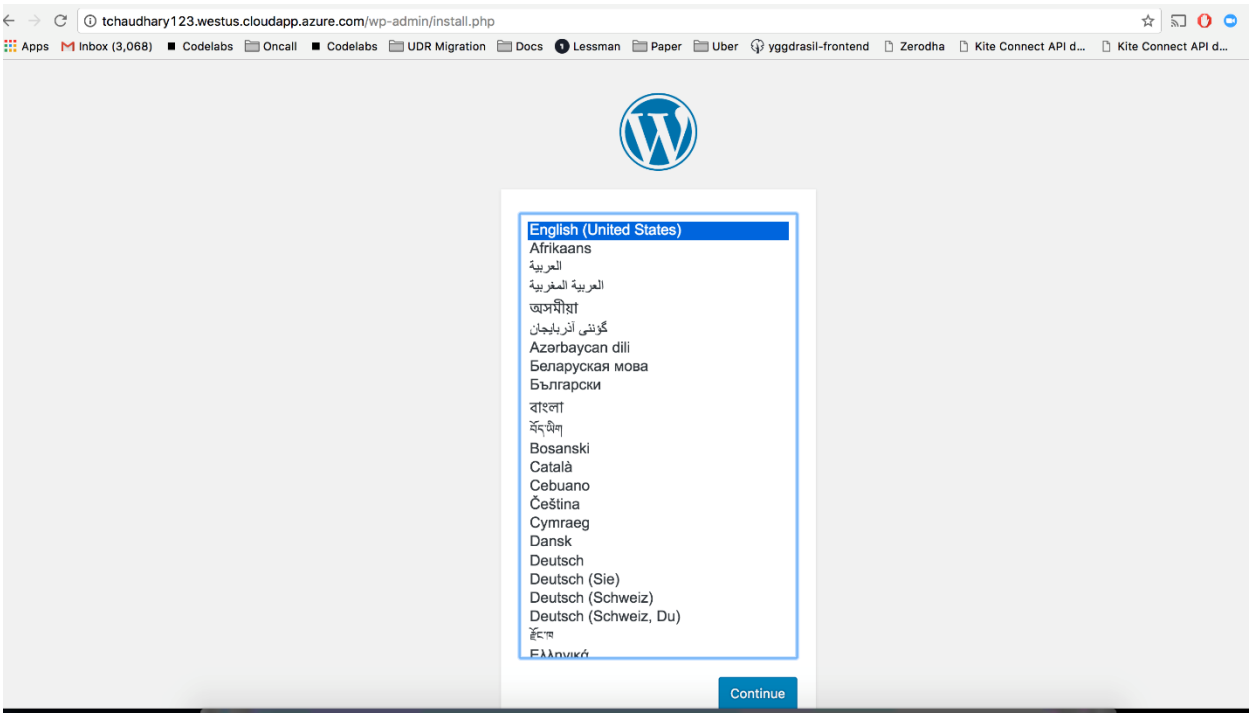
Step8: Verifying if the containers are up.

```
tchaudhary@MyDockerVM:~$ docker-compose ps
         Name                       Command               State         Ports
-------------------------------------------------------------------------------
tchaudhary_db_1          docker-entrypoint.sh mysqld      Up      3306/tcp
tchaudhary_wordpress_1   docker-entrypoint.sh apach ...   Up      0.0.0.0:80->80/tcp
tchaudhary@MyDockerVM:~$
```

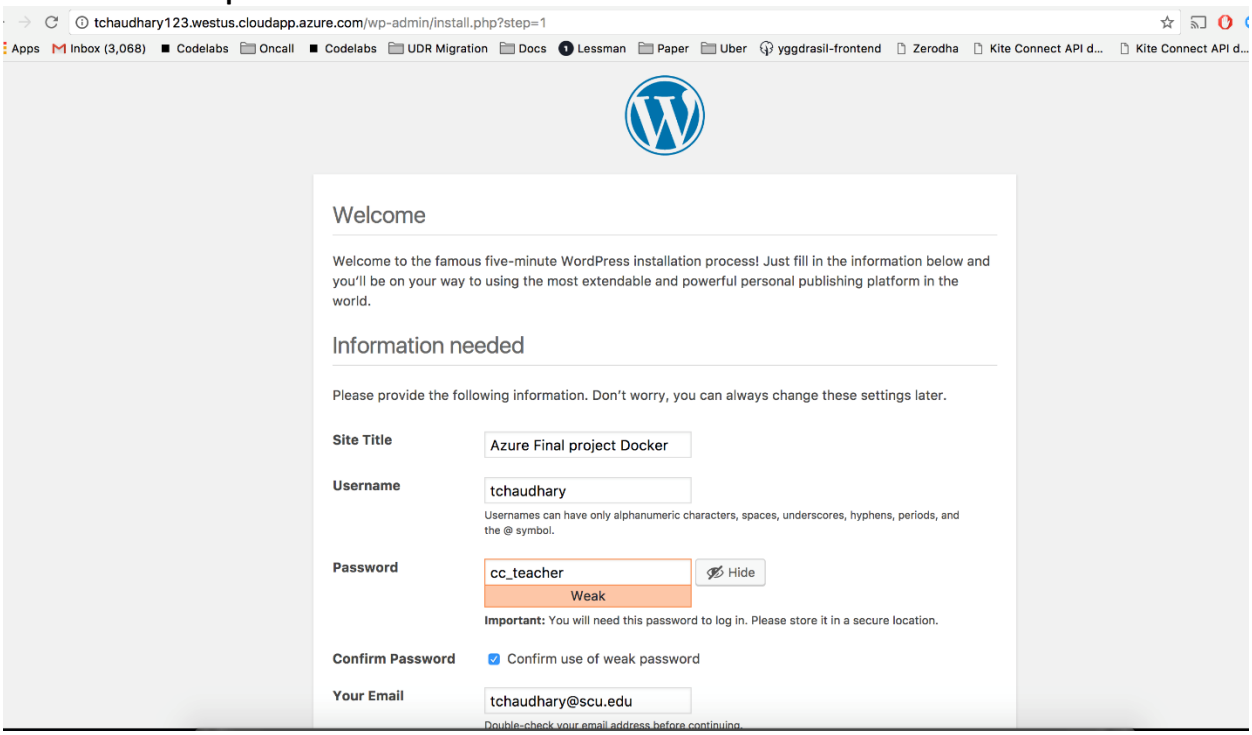
Step9: Docker VM is up on the Azure account!



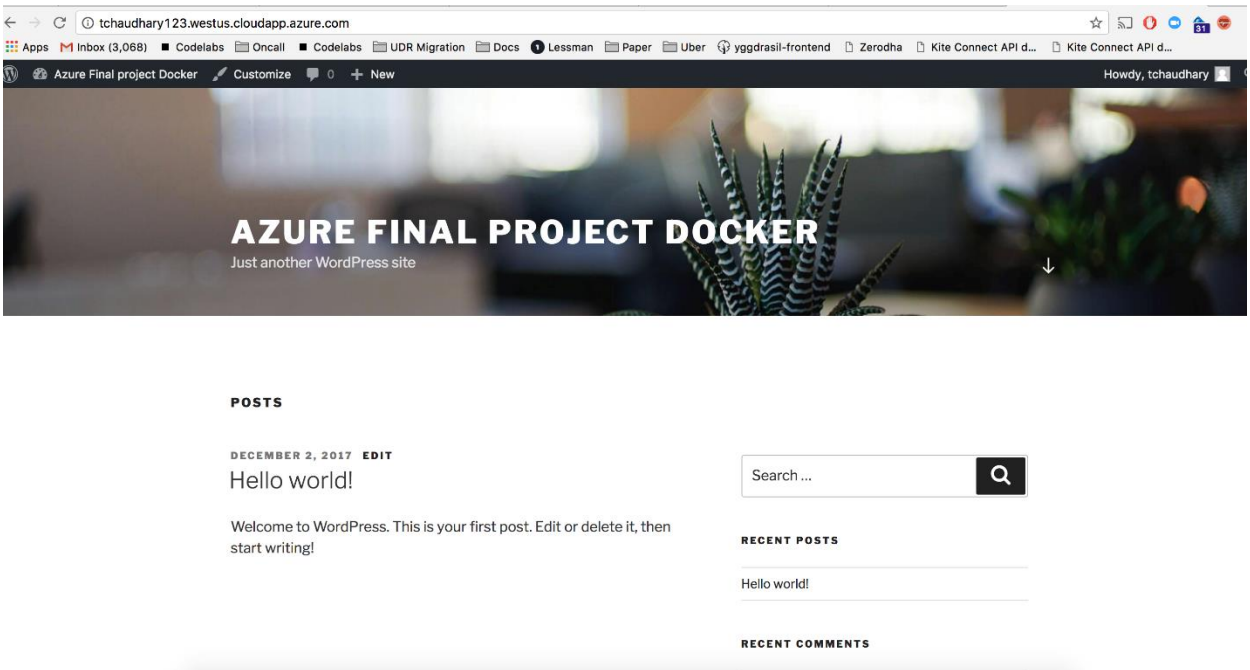
Step9: Connected to wordpress directly on port 80. Opened the browser and entered the DNS name of my VM i.e. http://tchaudhary123.westus.cloudapp.azure.com/



Installed the wordpress



Step10: My public DNS is displaying the wordpress blog
Link: http://tchaudhary123.westus.cloudapp.azure.com/

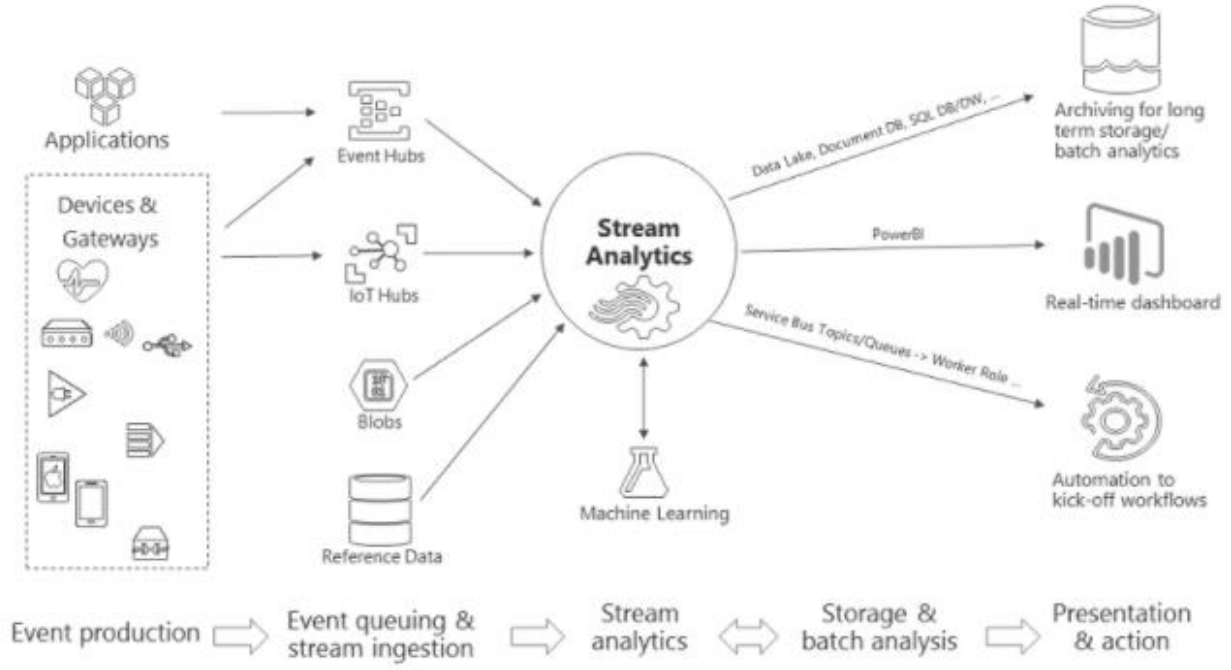


Part 4: BONUS

Introduction

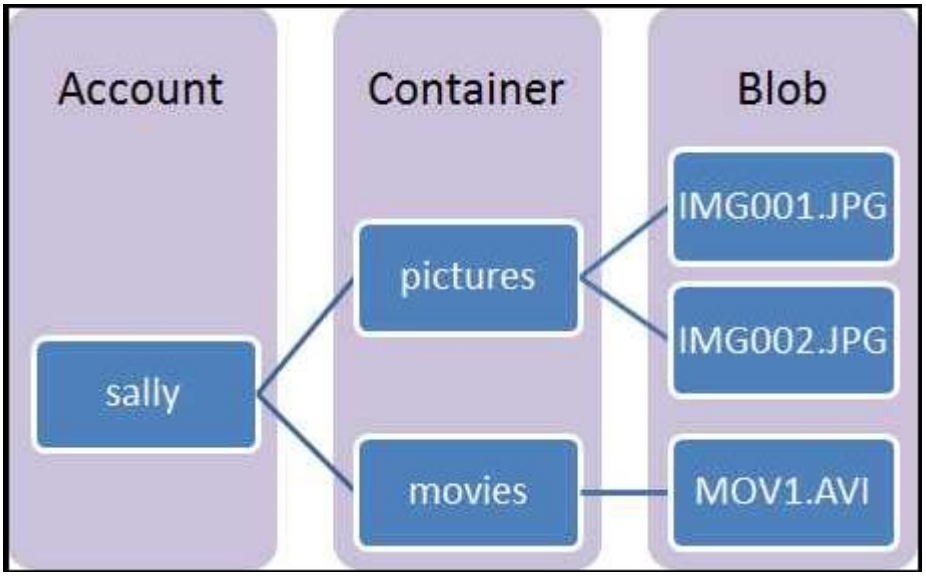
For a telecommunication company, Fraud costs them both in terms of loss in income and capacity wastage. It not only affects the network provider, but also the customer using the telecom services. Why most of the frauds happen in the telecommunication industry? The reason being calling through a mobile network is not concerned with the physical place the person is calling from and also a mobile subscription is not hard to get. This has become an illegal profit-making business with very little investment and less chances of getting caught. Catching a fraud would be nice but detecting a real-time fraud would be much helpful. So, for the purpose of the project, I will be taking a telecommunications company scenario who wants to detect real-time fraudulent calls.
Real-time analytics on data is becoming extremely valuable these days with the increasing amount of data generated on a daily-basis. Analyzing data to transform into trends, patterns and actionable insights has become a necessary thing. Multiple algorithms are run on the same data multiple times to enable fraud detection and not let any single fraudulent call to pass the test.
Microsoft Azure has a powerful service called Azure STREAM ANALYTICS which performs real-time analysis on streaming data and help us to identify the patterns and relationships within the data. The source of data can be devices, websites, social media or applications.

Once the output has been generated from the stream analytics, we can use another good feature of Microsoft Azure called POWER BI. This is one of the trending Business Intelligence tools that helps in transforming data into actionable reports. We can produce interactive visualizations out of the data and the data can be an excel sheet, csv, web pages, Google Analytics, Azure Analysis services, Azure SQL databases and many more. Below is the chart to show the working of the Stream Analytics and the detailed working will be explained by the steps taken to execute the project:



Key Problem & Cloud Solution

Taking into consideration the telecommunication industry, we all know they get large amount of data about incoming calls every day. In these incoming calls, we want to help the company to detect the real-time fraudulent calls and either act towards it or inform the customer about it. The SIM fraud detection criteria we have is getting multiple calls from the same number around same time, but all the calls are from places geographically dispersed. So, to identify such problem, we will use Azure's services like Eventhub to ingest the events, Stream Analytics job where a SQL-like query will be written to filter out the input data, Blob Storage to automatically store the output generated after each job stops running. **Blob Storage is S3** equivalent in Azure which is specialized to store objects or blobs. Every blob storage account has a container that stores the blobs that can be retrieved later for analysis.



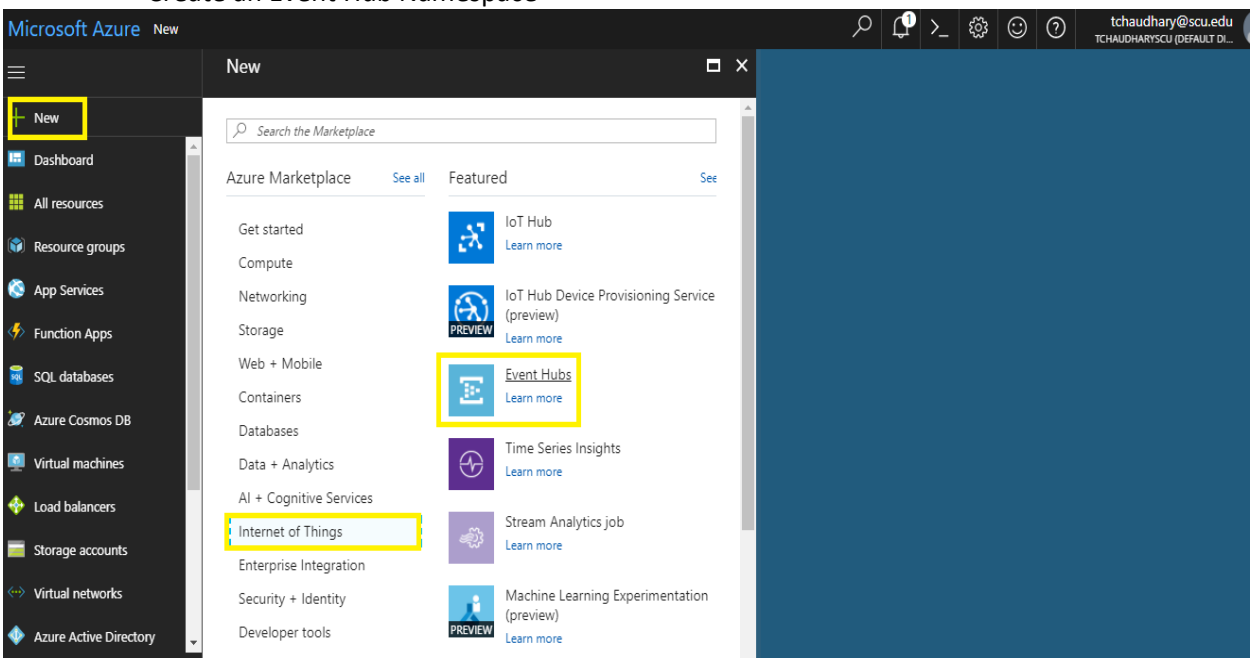
In our case, once the Stream analytics job stops running, the output i.e. information on fraudulent calls is stored as a json file in the container. I have also used another service of Azure called Power BI through which we can visualize the real-time data using line charts. But since one job can only have one SQL query, I could not specify multiple output sinks. So, I have tried and executed both Blob storage account (AWS S3 equivalent) and Power BI and have put the screenshots, and finally for the working demo, it is putting the generated output in Blob storage container. The EventHub requires real-time input data, and that data needs to be fed to the stream analytics job, so after logging into the account you can only see the running job with all the configuration and the json files stored in the blob storage container "fraudulentcalls" under the storage account "tchaudharyfinalproject". But to see the real-time data generation, we need to run the telco generator app to feed in the data. The name of the stream analytics job is: "Fraud-Detection-job"

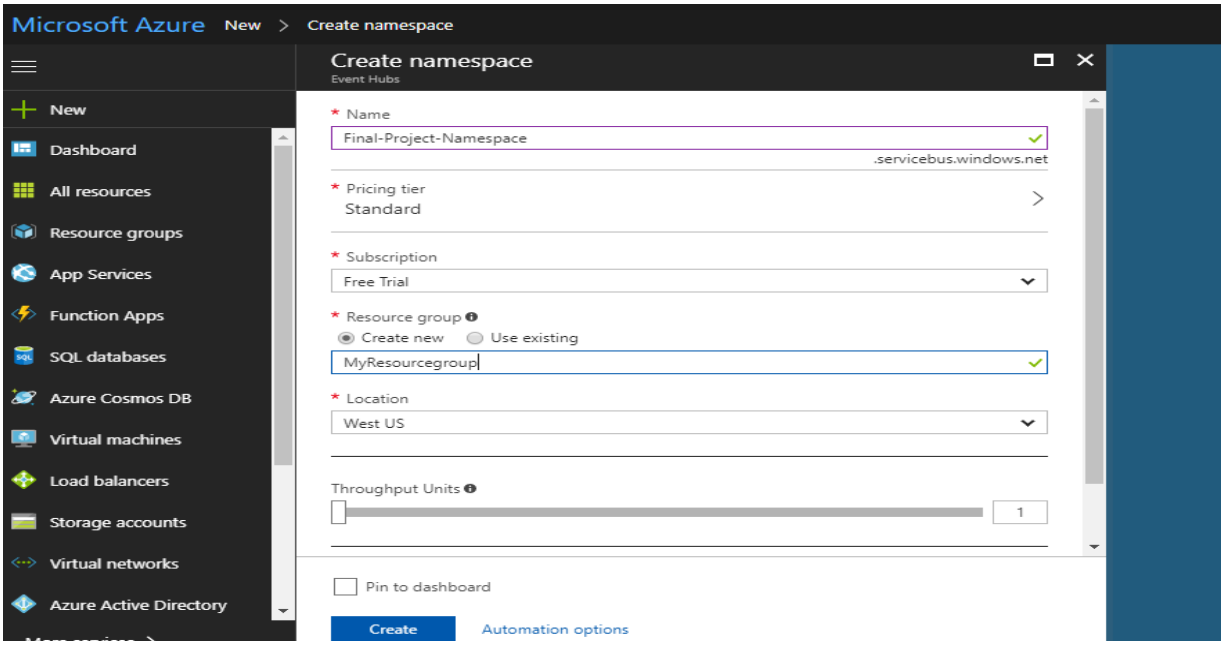
Azure credentials:
Username: tchaudhary@scu.edu
Password: cc_teacher

Process

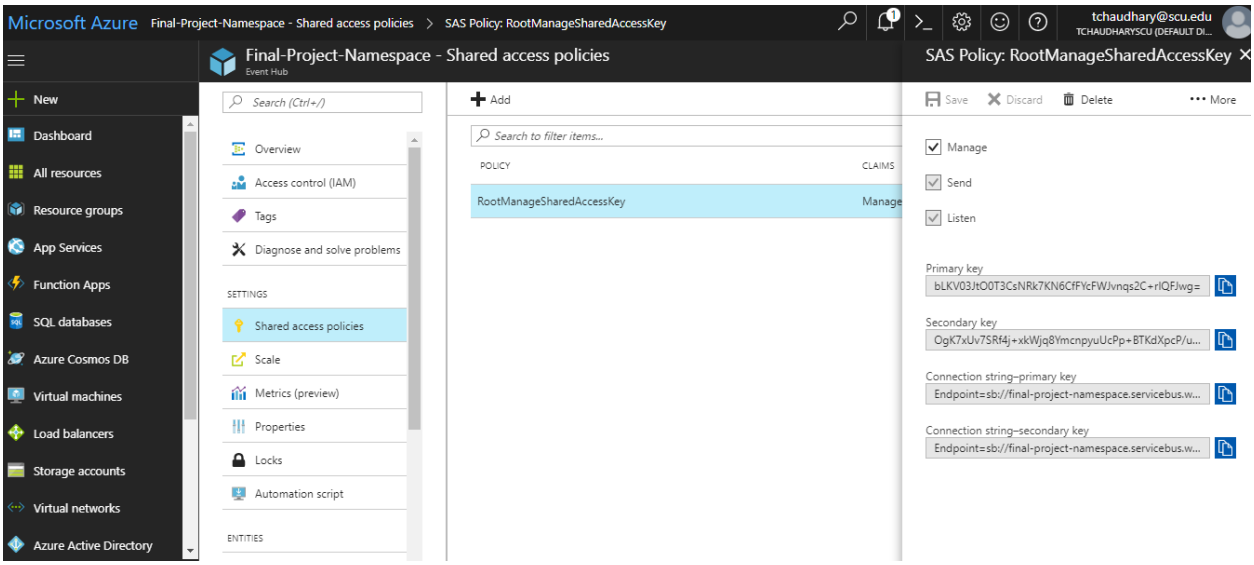
1. For analyzing the data stream, we will be ingesting the data in Azure using Azure Event Hubs which allows us to ingest large number of events per second and then store their information. So, here we have a sample phone call data generated by call-event generator app (will be discussed later) which will be sent to event hub.

    - Create an Event Hub Namespace

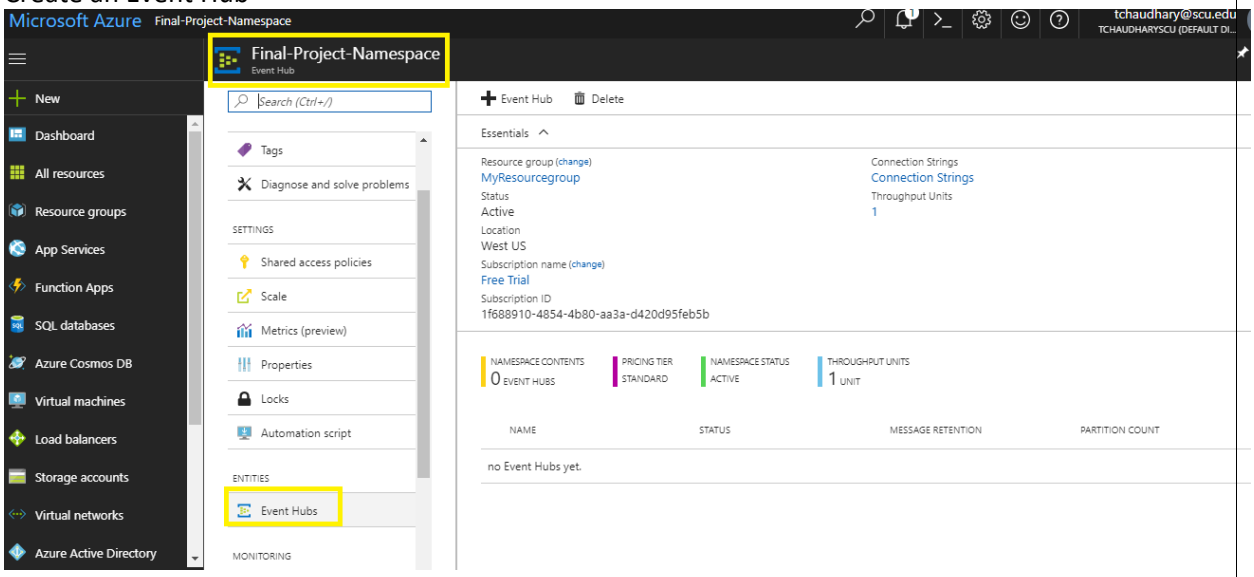




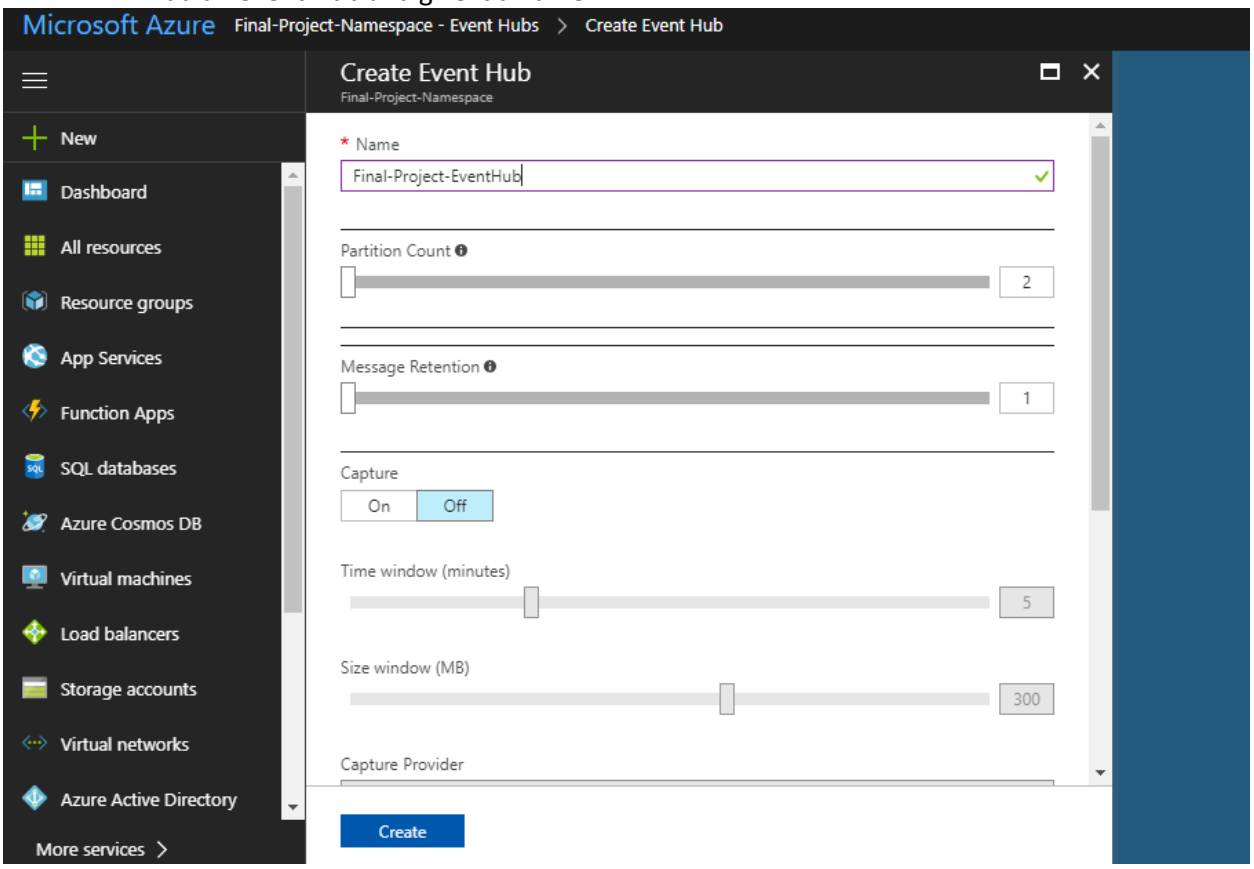
- For the namespace created, click shared access policies, then click on RootManageSharedAccessKey and copy the connection string for later use.



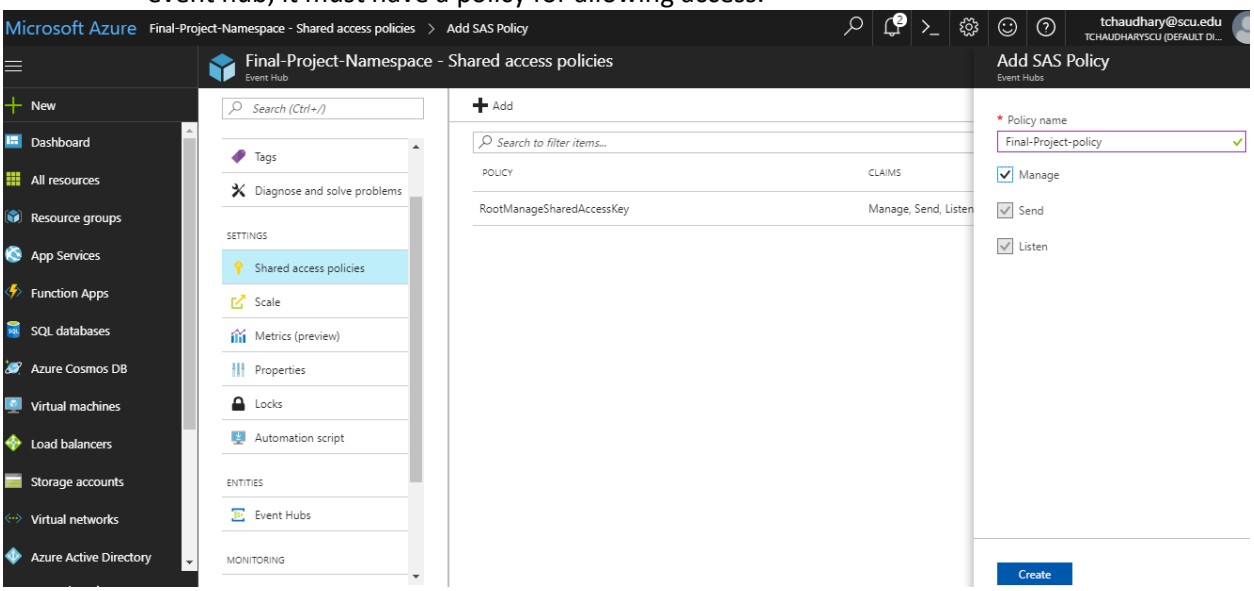
- Create an Event Hub



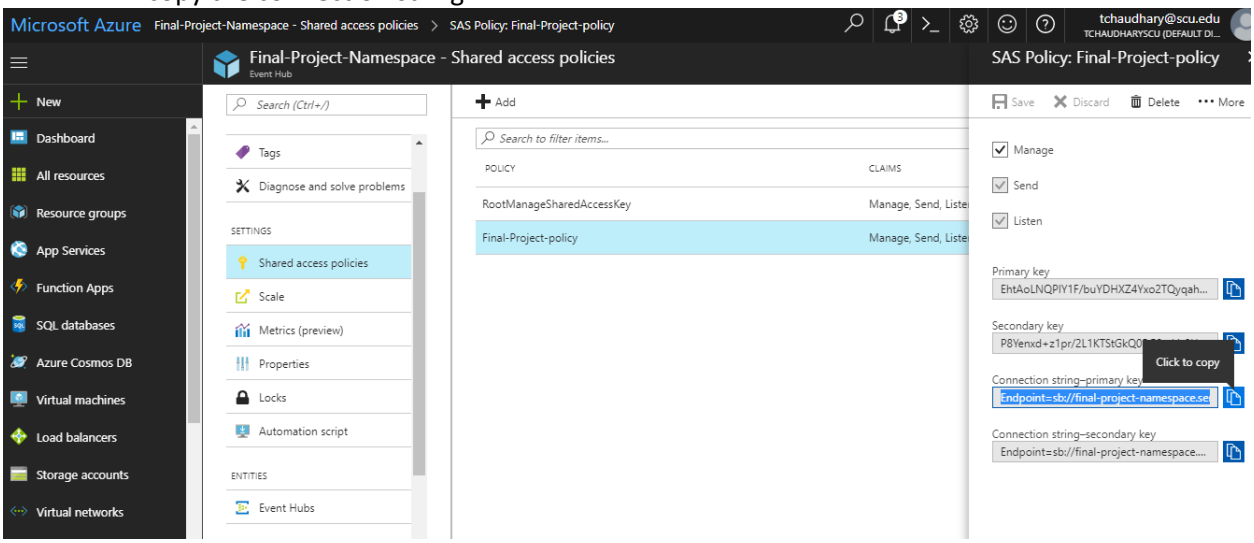
- Add an event Hub and give it a name



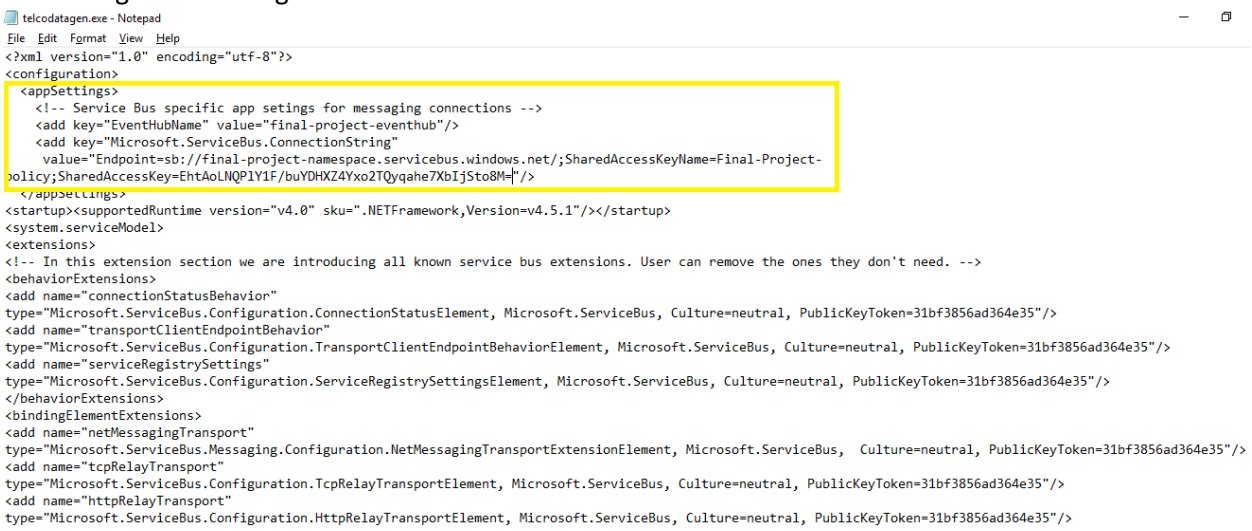
- Grant access to the Final-Project-EventHub and get the connection string. To send data to an event hub, it must have a policy for allowing access.



- Click on the newly generated shared access policy with the name Final-Project-policy and copy the connection string.



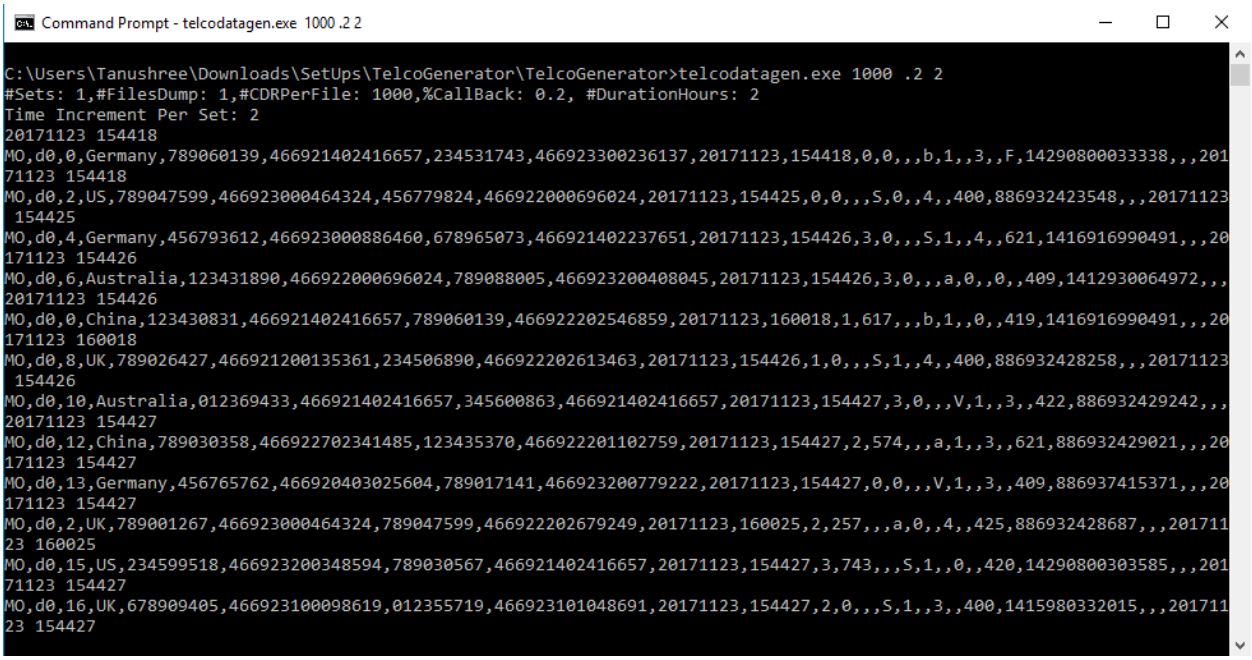
2. To send the call records to the event hub, configure and start the event generator application. Downloaded and unzipped the TelcoGenerator.zip file. Edit the app settings for telcodatagen.exe.config file and save it.



3. Using the command prompt, go to the folder where telco generator app was unzipped and type the command: `telcodatagen.exe 1000 .2 2`
   It has the following parameters:
   #CDRPerFile: 1000
   %CallBack: 0.2 tells that 20% call records will look fraudulent calls.
   #DurationHours: 2 says the no. of hours app will run.

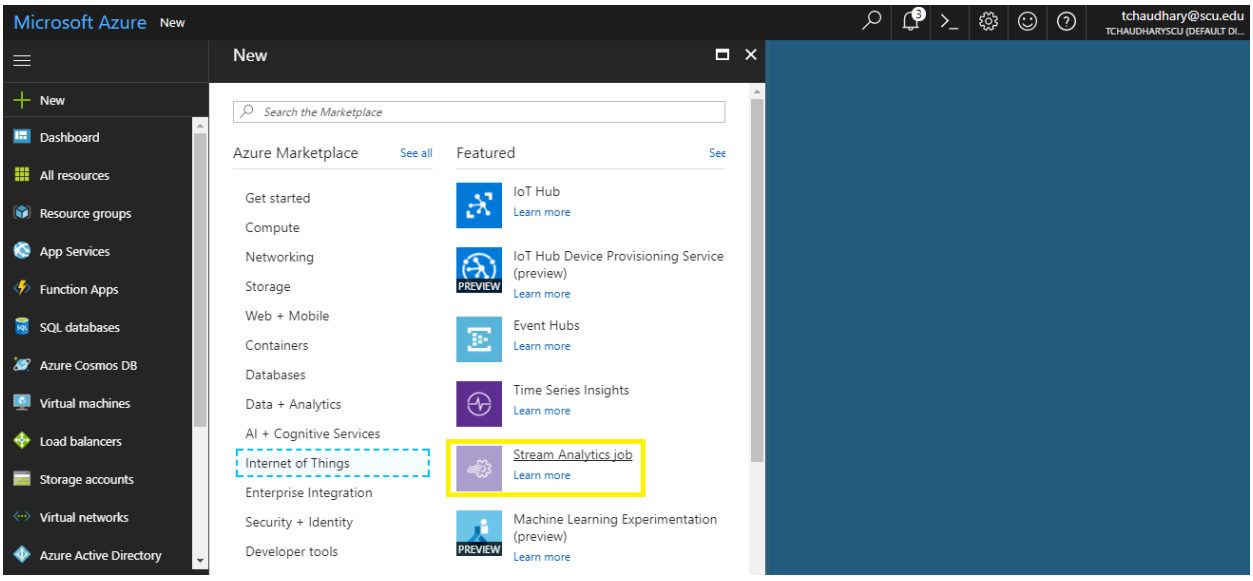
   Now the app is displaying the call records being sent to the event hub. The key fields as we see in the records are Timestamp for call start time, telephone switch used to connect the call,

caller's phone number, Unique identifier of the caller(IMSI), Recipient's phone number, Unique identifier of the Recipient(IMSI).

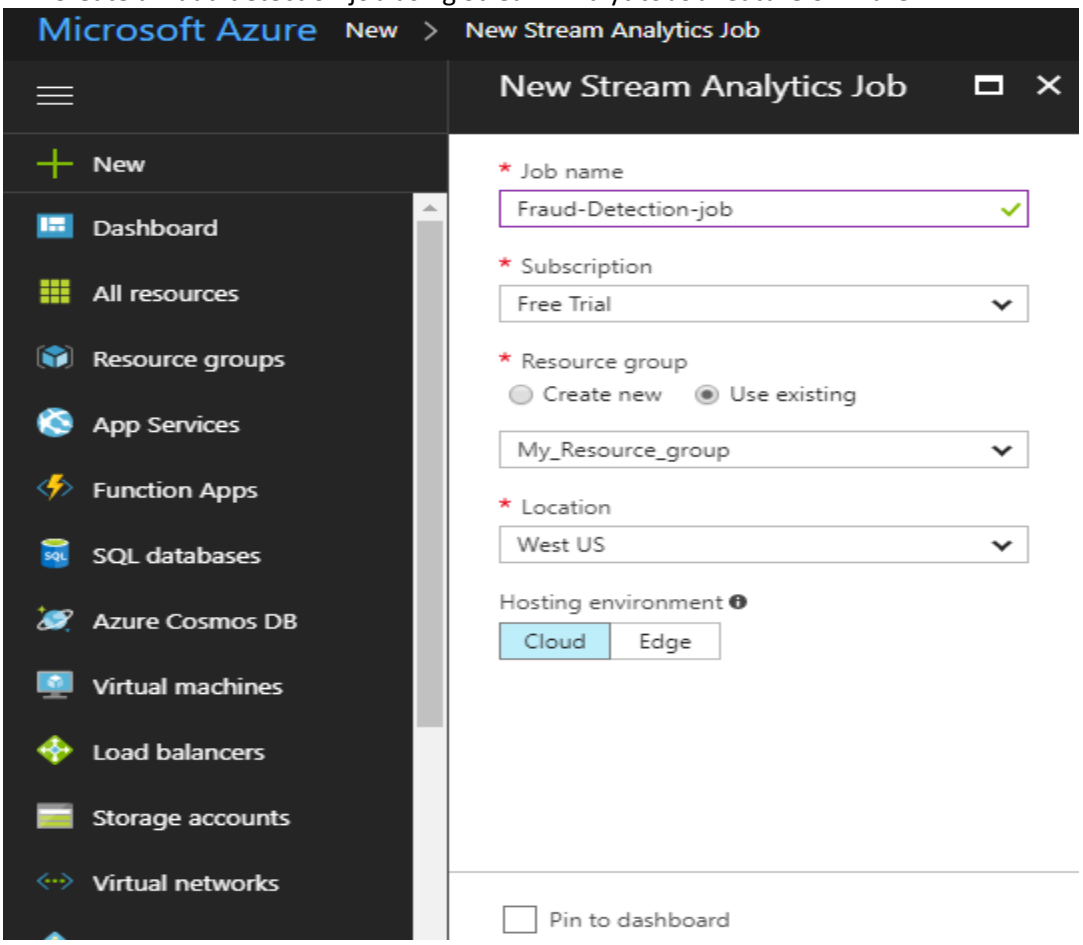


This app keeps running in the background and can be stopped using Ctrl+C.

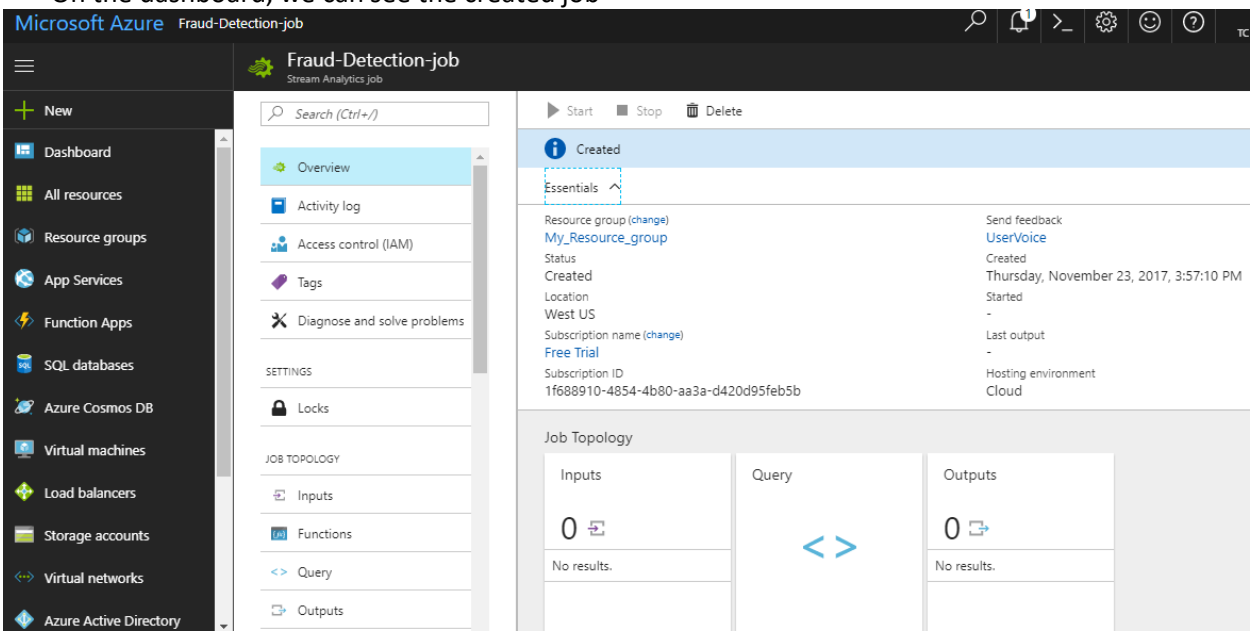
4. Now we need to set up the Azure stream Analytics job on the stream of call records being sent to the event hub.



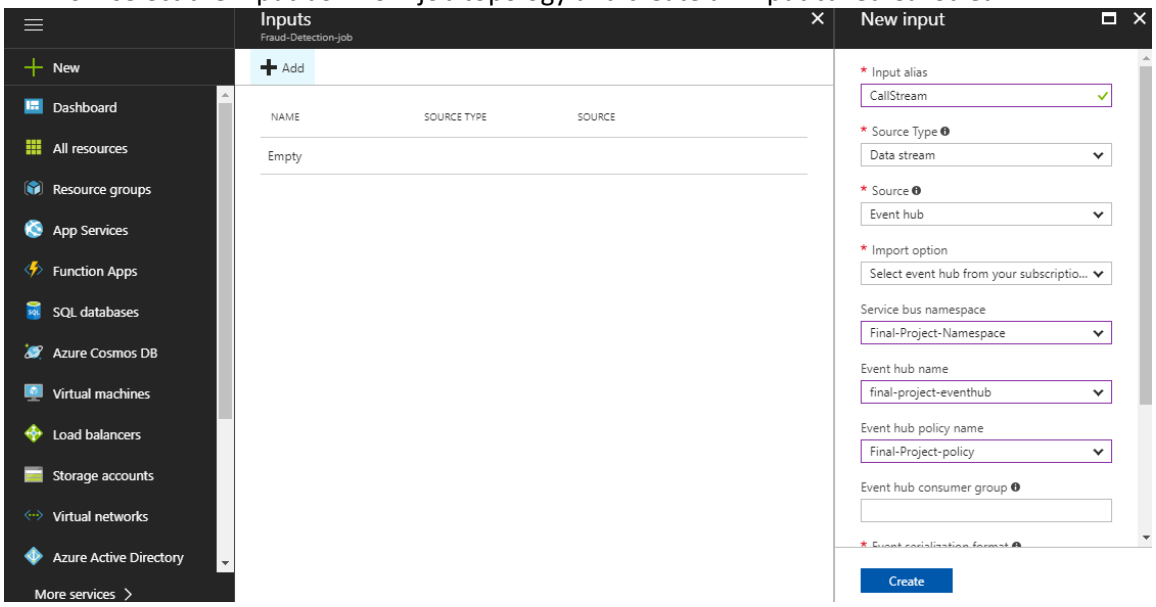
- Create a fraud detection job using Stream Analytics Job feature of Azure.



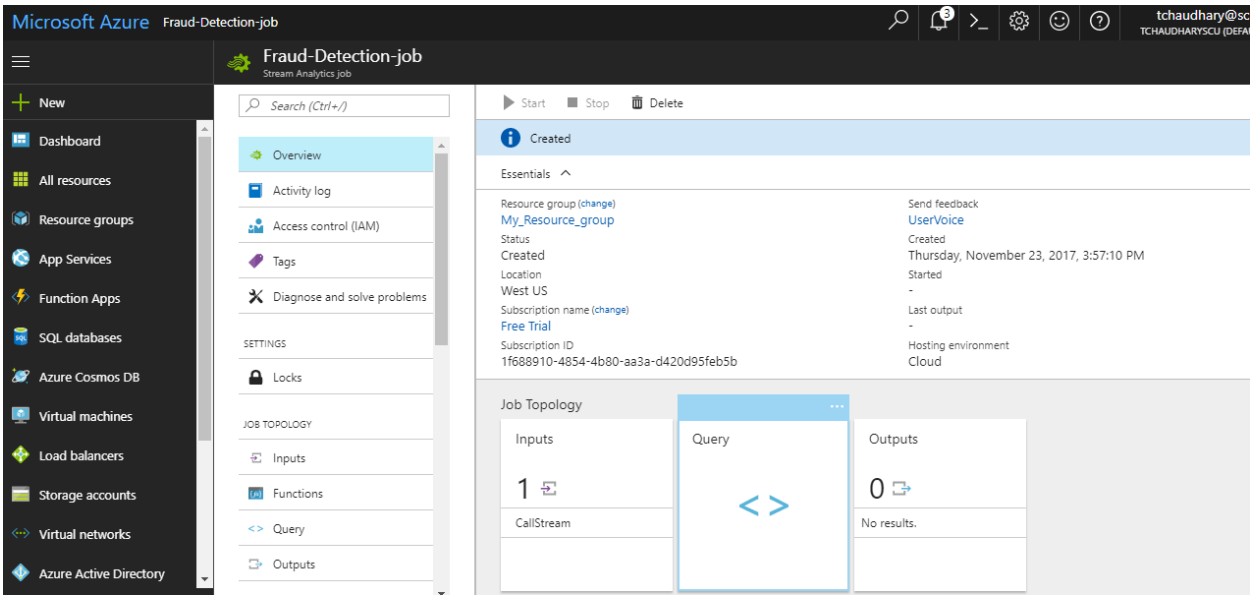
- On the dashboard, we can see the created job



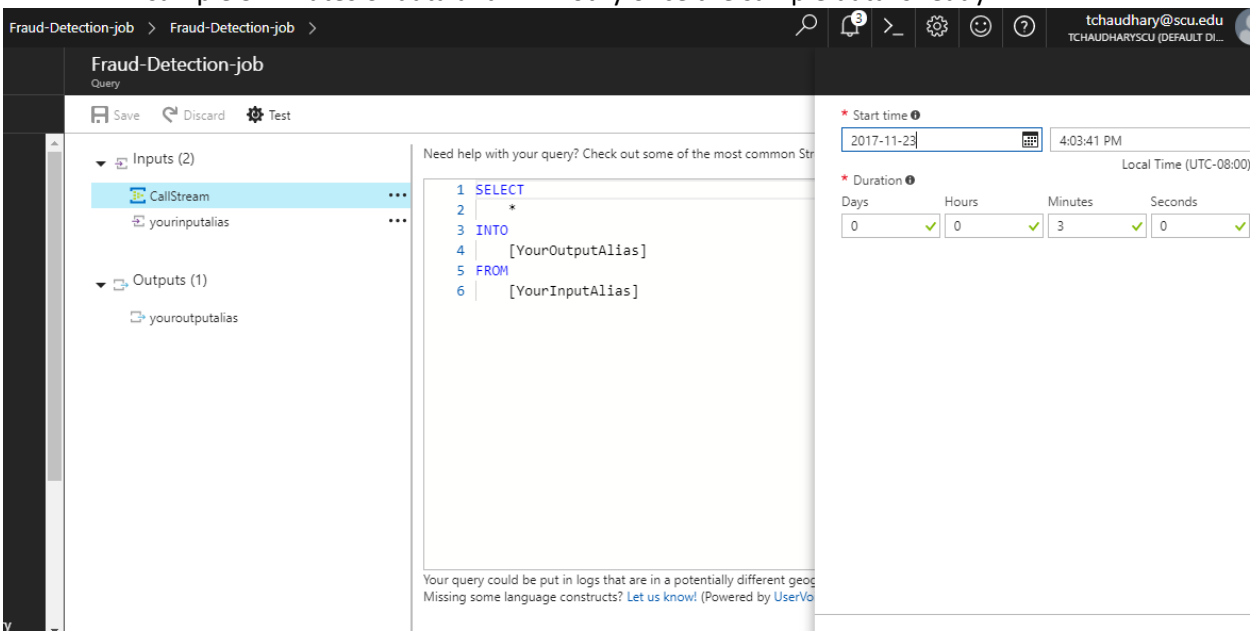
- Now select the input box from job topology and create an input called CallStream



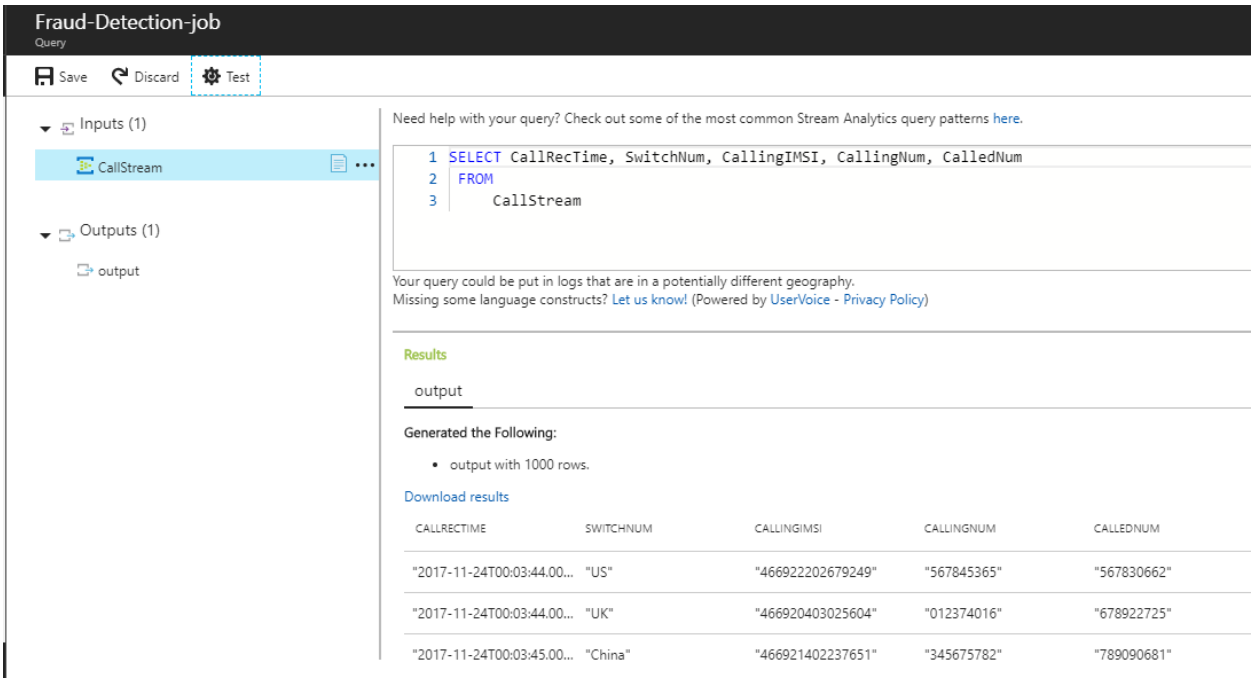
- Now click on Query box and create queries to transform data, the queries are in a SQL-like language with some specifications related to stream analytics.



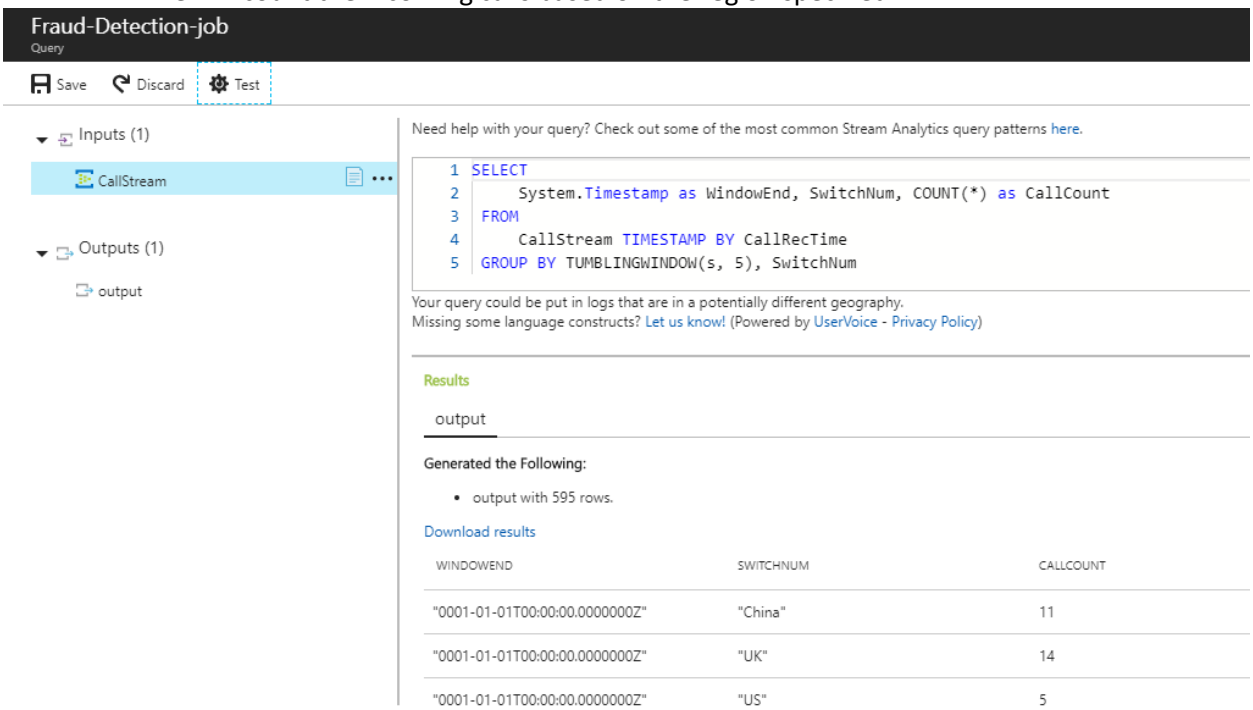
- We will upload the sample data and will put 3 minutes duration which means azure will sample 3 minutes of data and will notify once the sample data is ready.



- First query is to read some specific columns in the data. Write the query and click on Test to execute it.



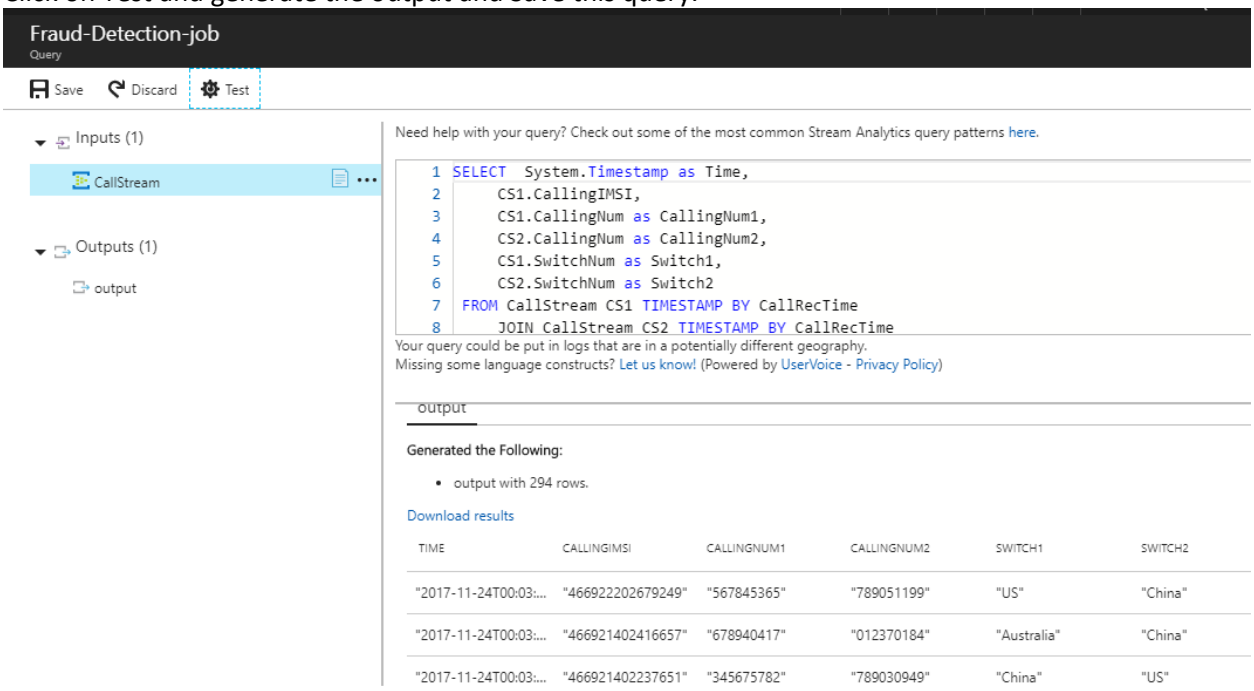
- We will count the incoming calls based on the region specified.



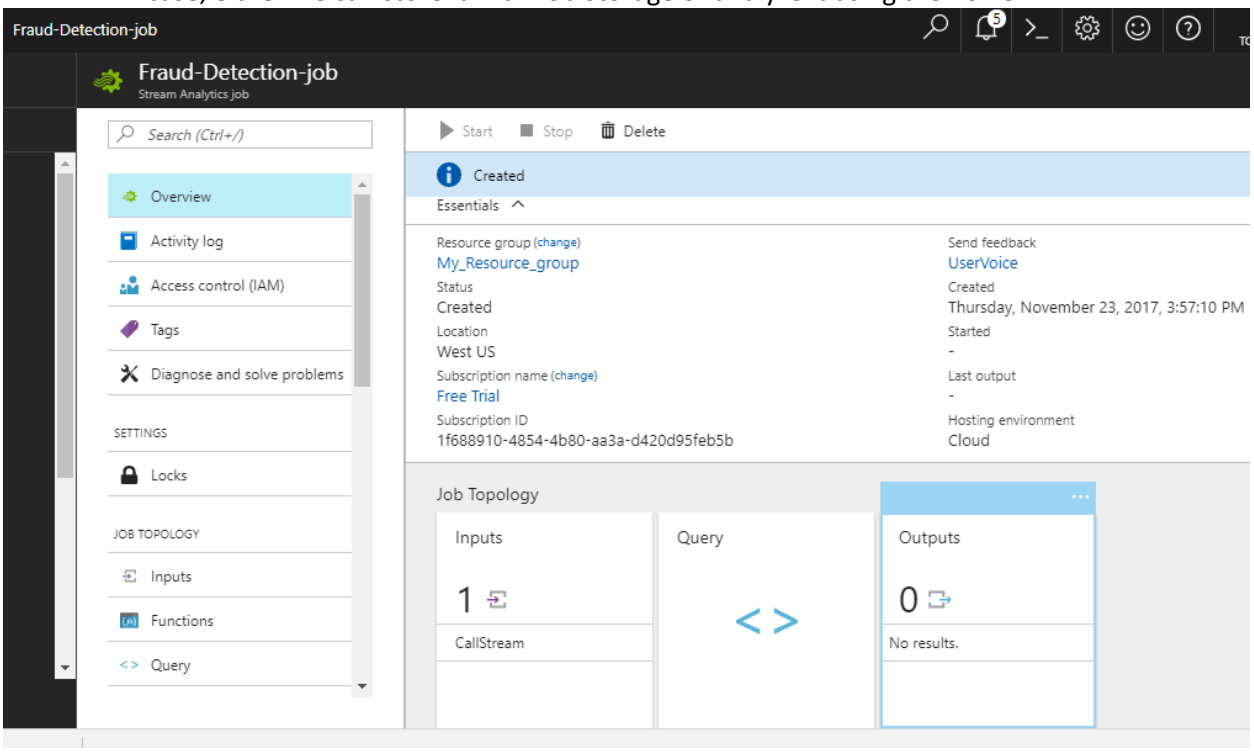
- Using a self-join based on CallRecTime, we will detect the SIM fraud. For example, the calls that come from the same user but from different locations within 5 sec.
  This is the query:

```
SELECT  System.Timestamp as Time,
        CS1.CallingIMSI,
        CS1.CallingNum as CallingNum1,
        CS2.CallingNum as CallingNum2,
        CS1.SwitchNum as Switch1,
        CS2.SwitchNum as Switch2
FROM CallStream CS1 TIMESTAMP BY CallRecTime
        JOIN CallStream CS2 TIMESTAMP BY CallRecTime
        ON CS1.CallingIMSI = CS2.CallingIMSI
        AND DATEDIFF(ss, CS1, CS2) BETWEEN 1 AND 5
WHERE CS1.SwitchNum != CS2.SwitchNum
```

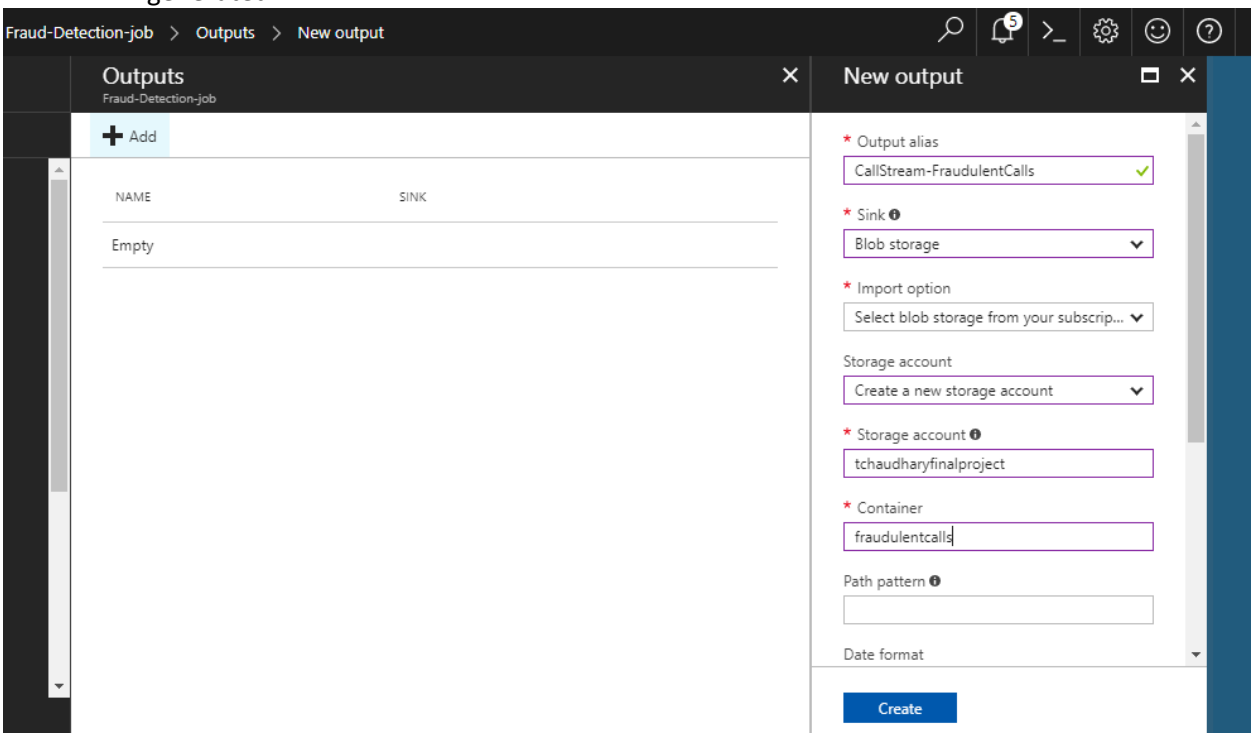
Click on Test and generate the output and Save this query.



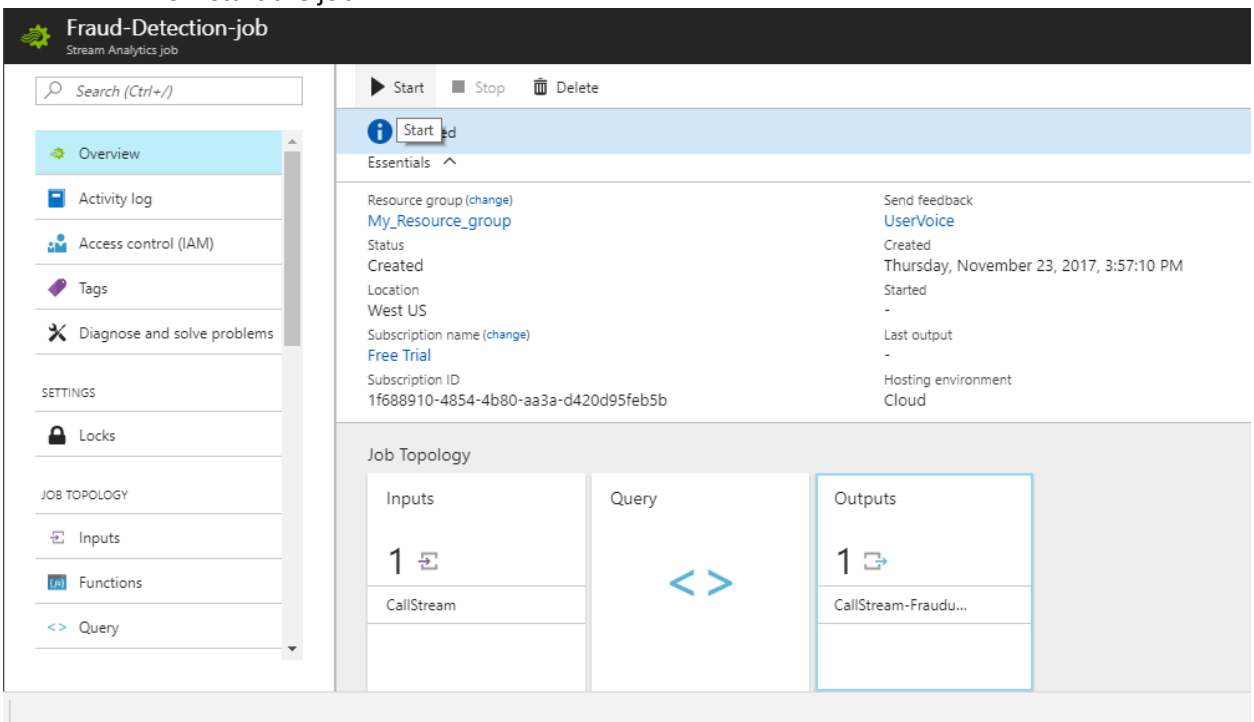
- Now the output generated needs to be stored somewhere. There are two options in our use case, either we can store it in a Blob Storage or analyze it using the Power BI.



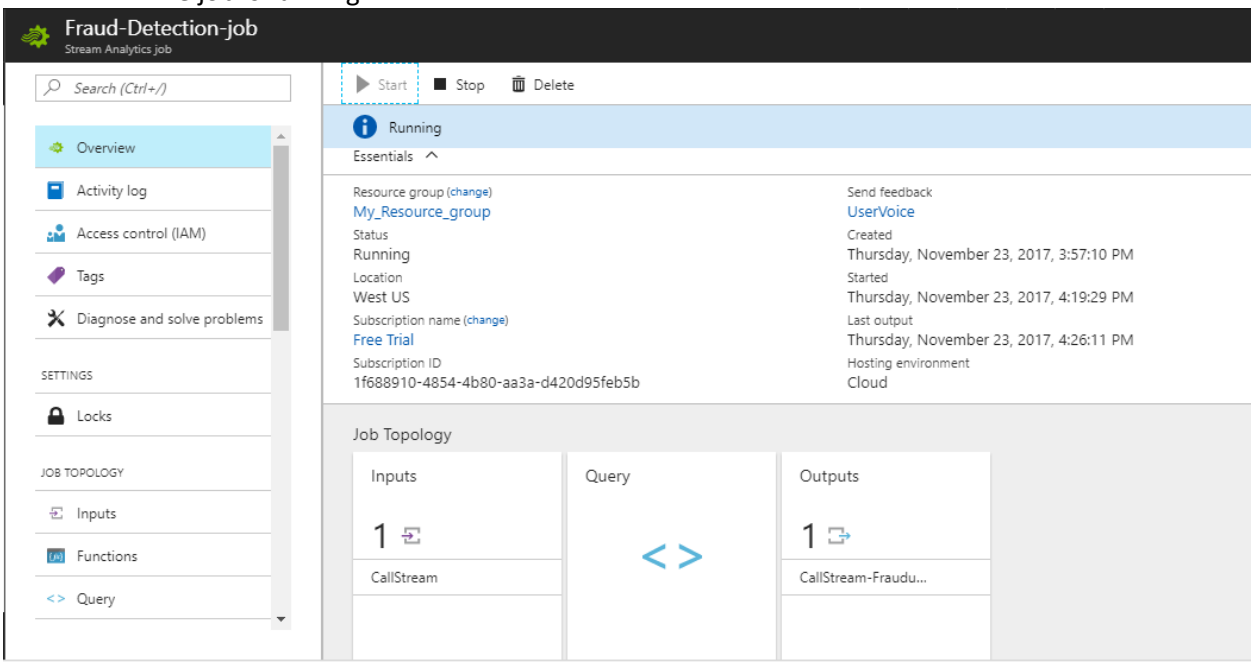
- Firstly, I will show creating a Blob Storage (AWS S3 equivalent) output sink to store the data generated.



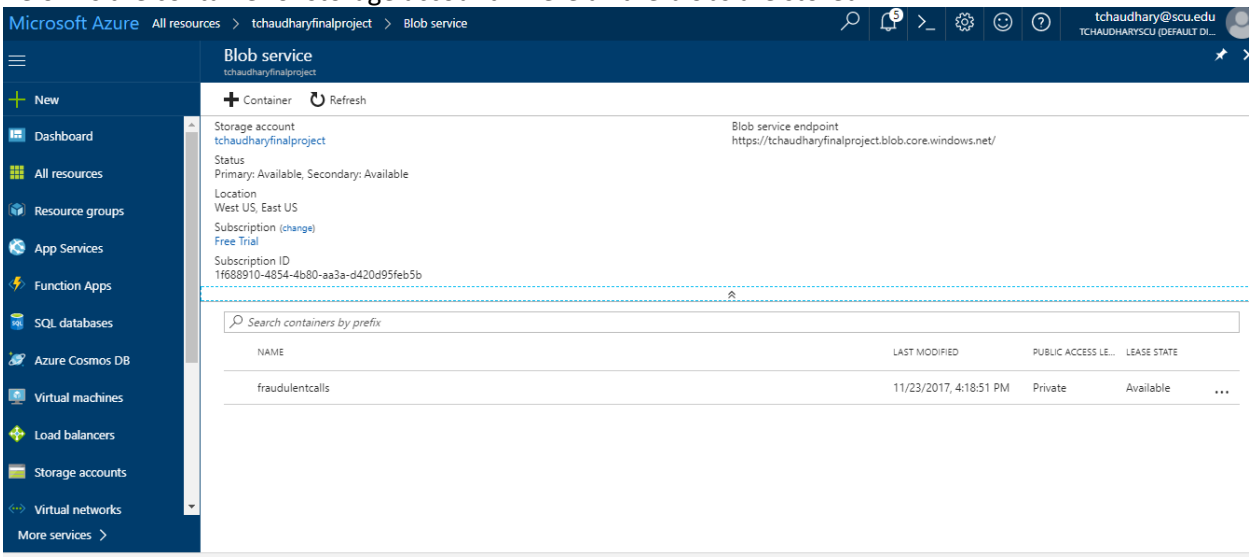
- Now start the job

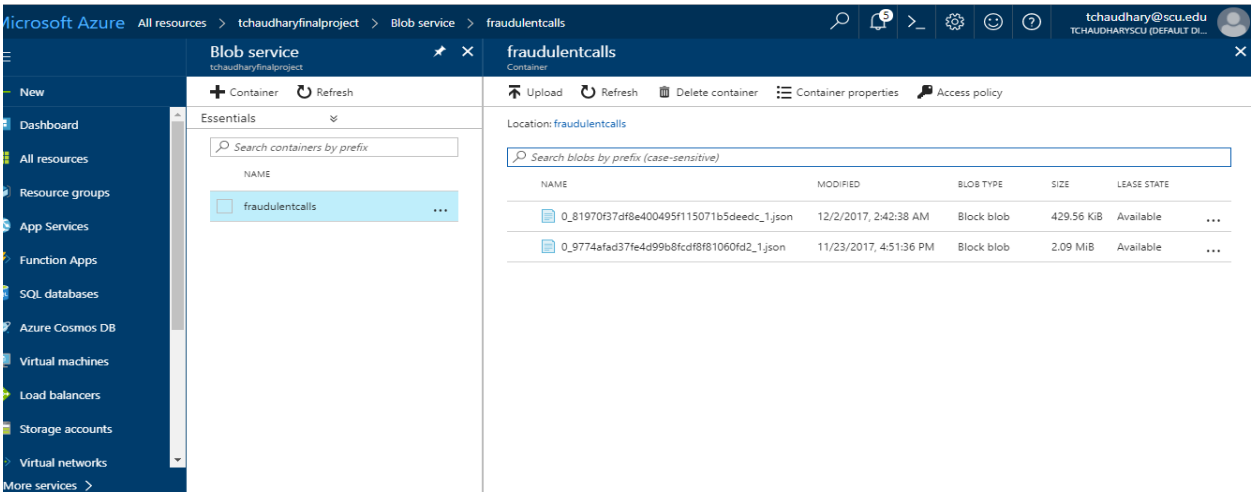


- The job is running.

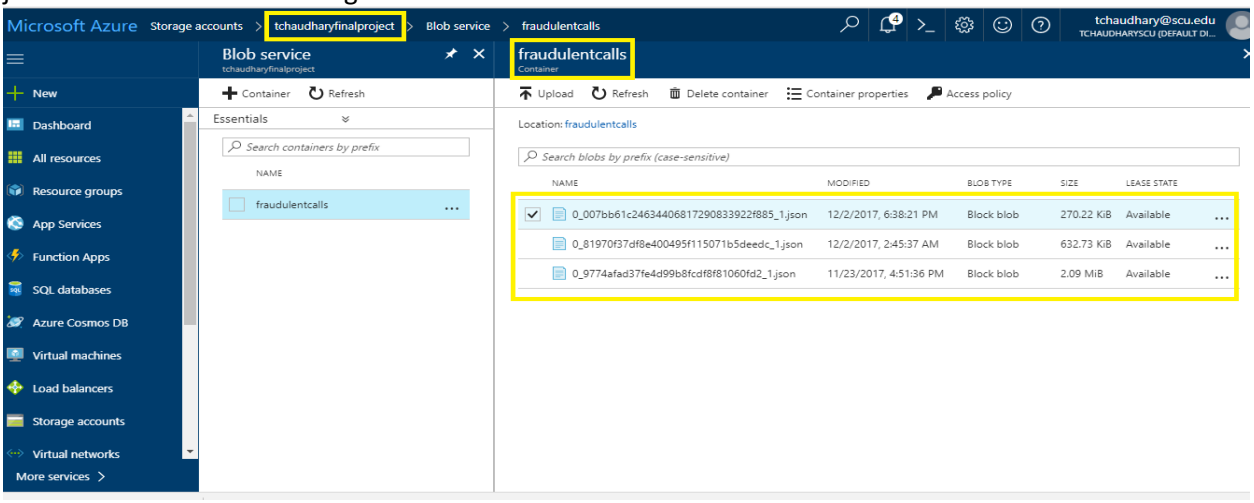


Below is the container of storage account where all the blobs are stored.

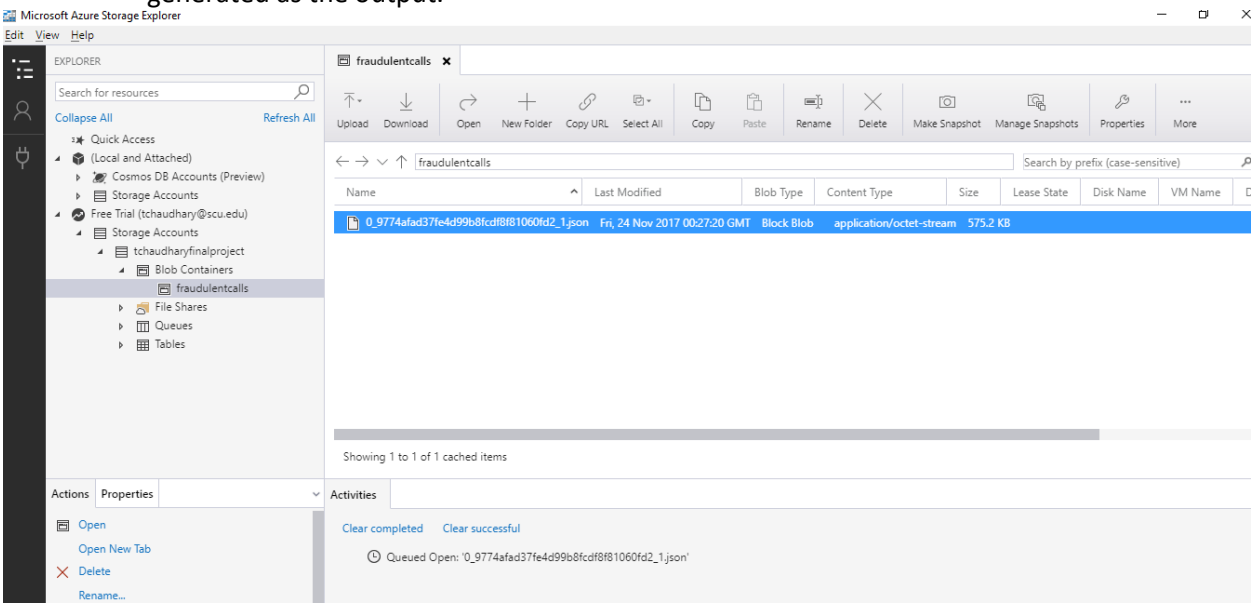




Highlighted are the names of storage account, container and the json files generated from running the jobs and stored in Blob storage.



- We will analyze the data using Microsoft Azure Storage explorer as well, open the json file generated as the output.



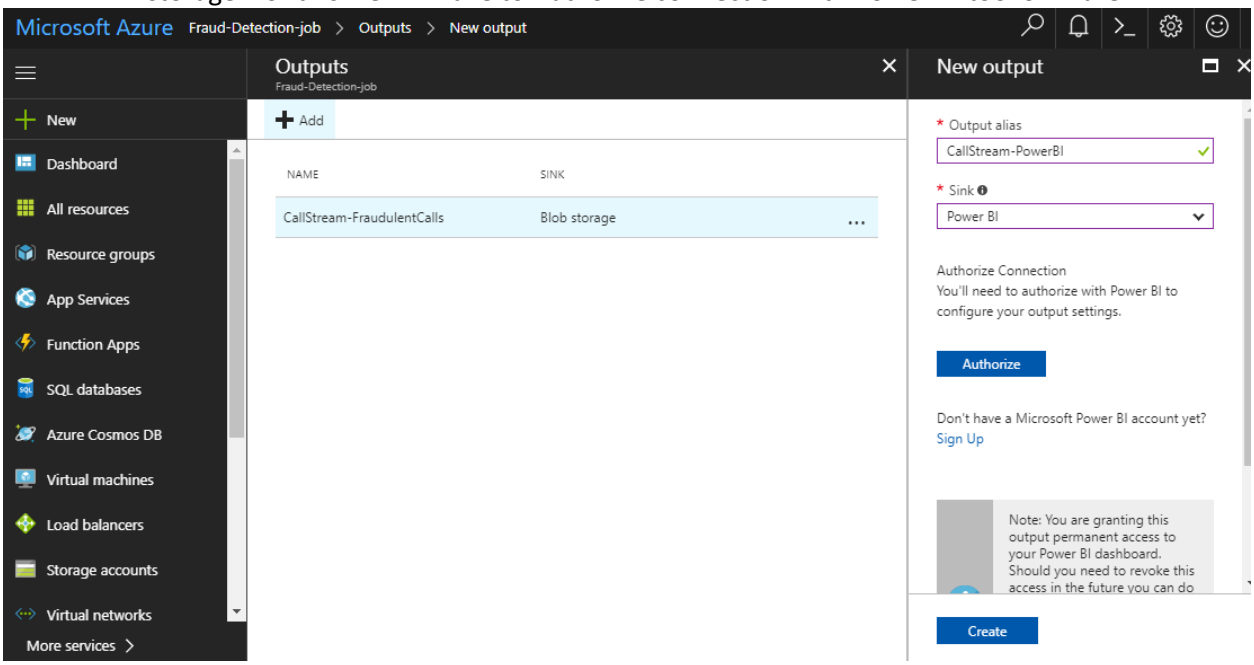
- And the output looks like this, these are the incoming fraudulent calls identified by the Stream Analytics. But this just gives us the idea that there are fraudulent calls but how much what is the running frequency. So, to further analyze the data we will use Power BI as output sink and then analyze the fraud calls.

0_9774afad37fe4d99b8fcdf8f81060fd2_1 - Notepad

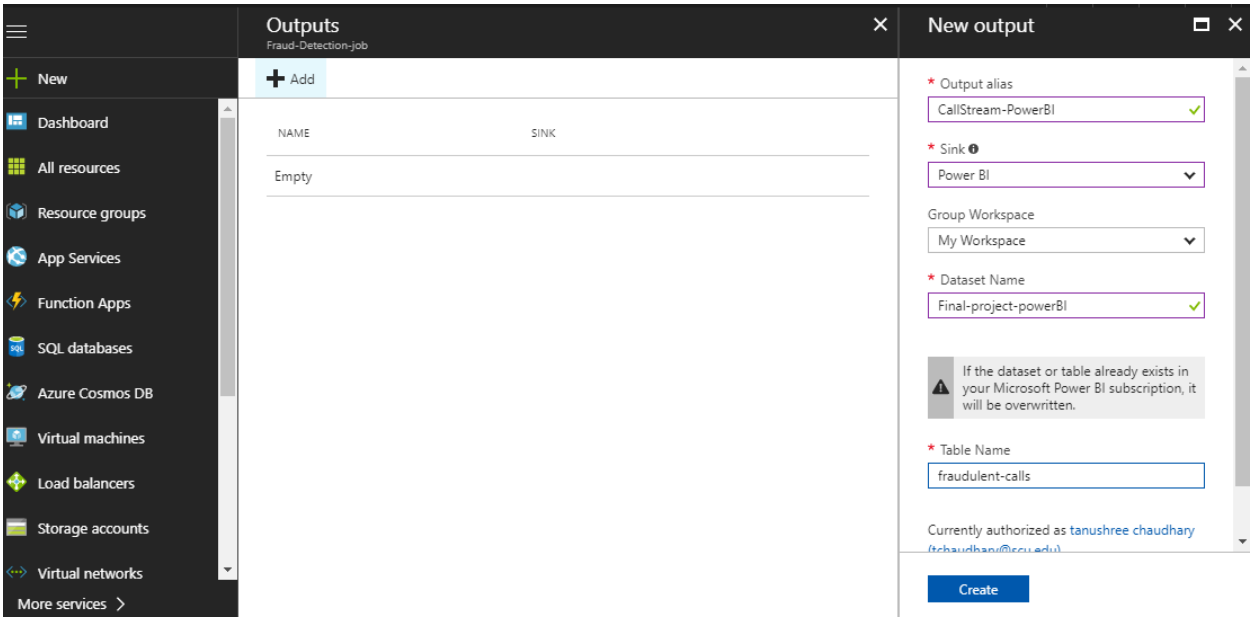
File Edit Format View Help

{"time":"2017-11-24T00:19:33.2180000Z","callingimsi":"466923200408045","callingnum1":"567867313","callingnum2":"345641797","switch1":"UK","switch2":"Australia"}
{"time":"2017-11-24T00:20:27.2440000Z","callingimsi":"466923300507919","callingnum1":"345643554","callingnum2":"789054002","switch1":"China","switch2":"Germany"}
{"time":"2017-11-24T00:20:57.3980000Z","callingimsi":"466920401237309","callingnum1":"345606678","callingnum2":"567839808","switch1":"Germany","switch2":"China"}
{"time":"2017-11-24T00:21:04.1990000Z","callingimsi":"466921402416657","callingnum1":"012324609","callingnum2":"678922757","switch1":"China","switch2":"Germany"}
{"time":"2017-11-24T00:21:05.4160000Z","callingimsi":"466923300236137","callingnum1":"345641809","callingnum2":"567898351","switch1":"Australia","switch2":"Germany"}
{"time":"2017-11-24T00:21:05.5720000Z","callingimsi":"466923000886460","callingnum1":"123494137","callingnum2":"567862562","switch1":"China","switch2":"US"}
{"time":"2017-11-24T00:21:05.5720000Z","callingimsi":"466923000886460","callingnum1":"789073823","callingnum2":"567862562","switch1":"UK","switch2":"US"}
{"time":"2017-11-24T00:21:05.6700000Z","callingimsi":"466921402237651","callingnum1":"234592012","callingnum2":"456791342","switch1":"Australia","switch2":"UK"}
{"time":"2017-11-24T00:21:05.6700000Z","callingimsi":"466921402237651","callingnum1":"789081491","callingnum2":"456791342","switch1":"Australia","switch2":"UK"}
{"time":"2017-11-24T00:21:05.6700000Z","callingimsi":"466921402237651","callingnum1":"789024021","callingnum2":"456791342","switch1":"Australia","switch2":"UK"}
{"time":"2017-11-24T00:21:05.9470000Z","callingimsi":"466923200408045","callingnum1":"789004354","callingnum2":"567872269","switch1":"US","switch2":"Germany"}
{"time":"2017-11-24T00:21:06.2440000Z","callingimsi":"466923200408045","callingnum1":"567872269","callingnum2":"456714393","switch1":"Germany","switch2":"US"}
{"time":"2017-11-24T00:21:06.3530000Z","callingimsi":"466922702346260","callingnum1":"678962558","callingnum2":"234577835","switch1":"UK","switch2":"US"}
{"time":"2017-11-24T00:21:06.3530000Z","callingimsi":"466922202613463","callingnum1":"012381667","callingnum2":"345686861","switch1":"UK","switch2":"China"}
{"time":"2017-11-24T00:21:06.3530000Z","callingimsi":"466922202613463","callingnum1":"234517149","callingnum2":"345686861","switch1":"UK","switch2":"China"}
{"time":"2017-11-24T00:21:06.8270000Z","callingimsi":"466921402416657","callingnum1":"678922757","callingnum2":"678971477","switch1":"Germany","switch2":"UK"}
{"time":"2017-11-24T00:21:06.9160000Z","callingimsi":"466920403025604","callingnum1":"789012089","callingnum2":"123400032","switch1":"UK","switch2":"China"}
{"time":"2017-11-24T00:21:07.2330000Z","callingimsi":"466920401237309","callingnum1":"678945730","callingnum2":"012319558","switch1":"China","switch2":"Germany"}
{"time":"2017-11-24T00:21:07.3890000Z","callingimsi":"466922200432822","callingnum1":"567872402","callingnum2":"567884488","switch1":"UK","switch2":"Germany"}
{"time":"2017-11-24T00:21:07.4470000Z","callingimsi":"466923200348594","callingnum1":"123464937","callingnum2":"234573613","switch1":"Germany","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.4470000Z","callingimsi":"466923200348594","callingnum1":"123430742","callingnum2":"234573613","switch1":"US","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.4470000Z","callingimsi":"466923200348594","callingnum1":"345652532","callingnum2":"234573613","switch1":"US","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.5090000Z","callingimsi":"466920400352400","callingnum1":"789076093","callingnum2":"456762147","switch1":"US","switch2":"China"}
{"time":"2017-11-24T00:21:07.5090000Z","callingimsi":"466920400352400","callingnum1":"345622263","callingnum2":"456762147","switch1":"US","switch2":"China"}
{"time":"2017-11-24T00:21:07.6340000Z","callingimsi":"466923000886460","callingnum1":"567862562","callingnum2":"123470728","switch1":"US","switch2":"China"}
{"time":"2017-11-24T00:21:07.6340000Z","callingimsi":"466923000886460","callingnum1":"789073823","callingnum2":"123470728","switch1":"UK","switch2":"China"}
{"time":"2017-11-24T00:21:07.6550000Z","callingimsi":"466920401237309","callingnum1":"678945730","callingnum2":"456752632","switch1":"China","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.7440000Z","callingimsi":"466923200348594","callingnum1":"123464937","callingnum2":"567857250","switch1":"Germany","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.7440000Z","callingimsi":"466923200348594","callingnum1":"123430742","callingnum2":"567857250","switch1":"US","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.7440000Z","callingimsi":"466923200348594","callingnum1":"345652532","callingnum2":"567857250","switch1":"US","switch2":"Australia"}
{"time":"2017-11-24T00:21:07.9780000Z","callingimsi":"466922702346260","callingnum1":"678962558","callingnum2":"234587074","switch1":"UK","switch2":"US"}

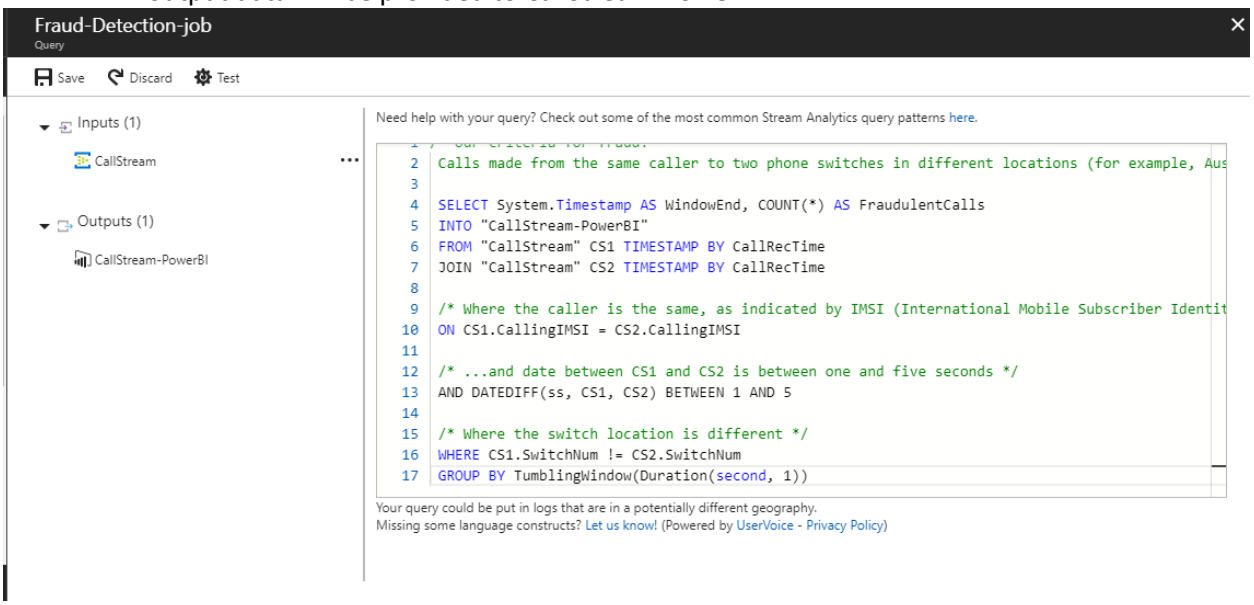
- So, we will stop the running job and change the output sink to Power BI instead of Blob storage. For this we will have to Authorize connection with Power BI tool of Azure.



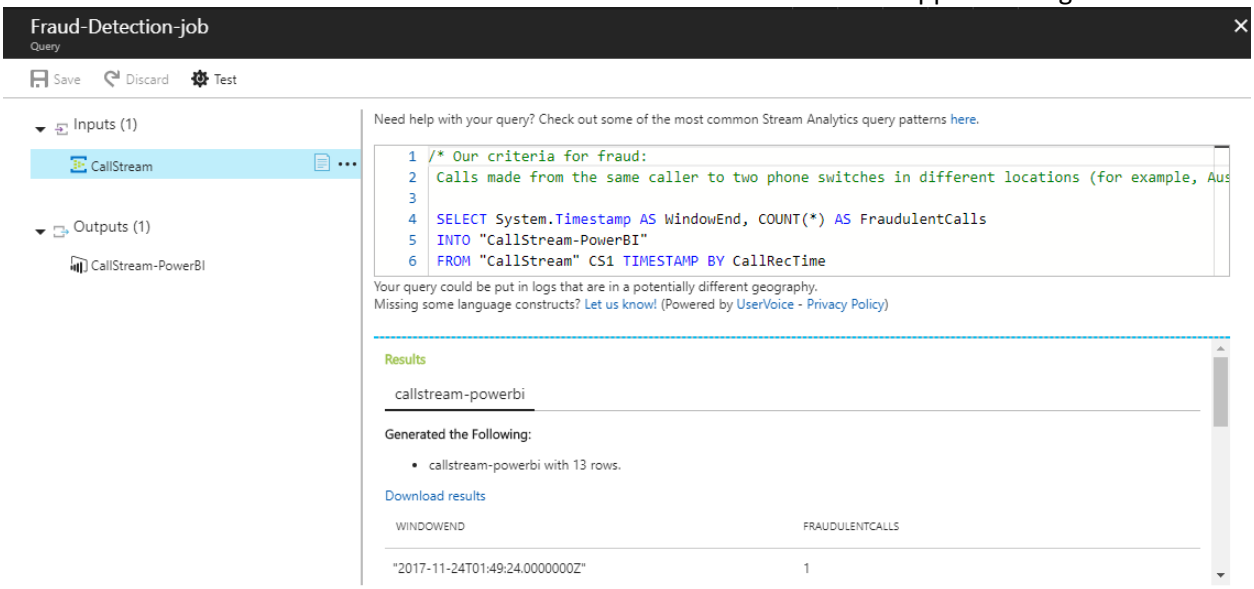
- Create the Data set name once the connection has been authorized.



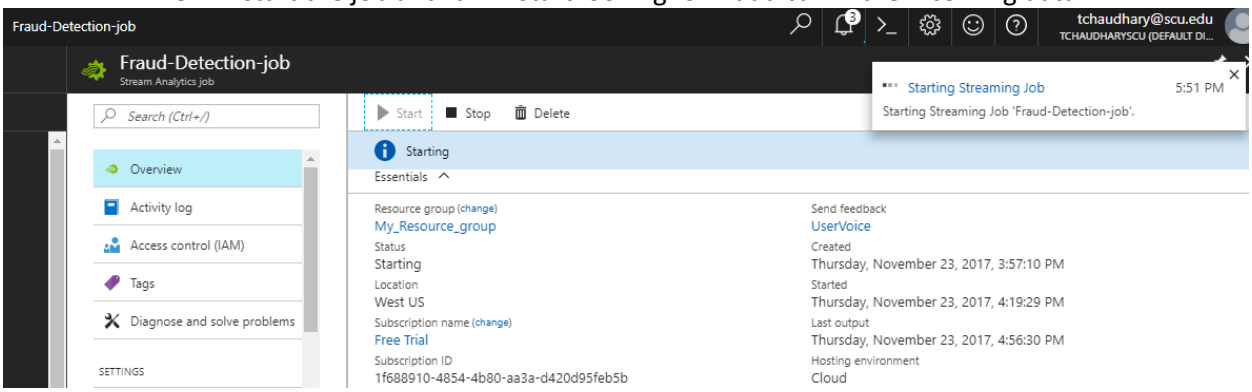
- In the query box, we will be writing a self-join query similar to the earlier one but this time output data will be provided to CallStream-PowerBI.



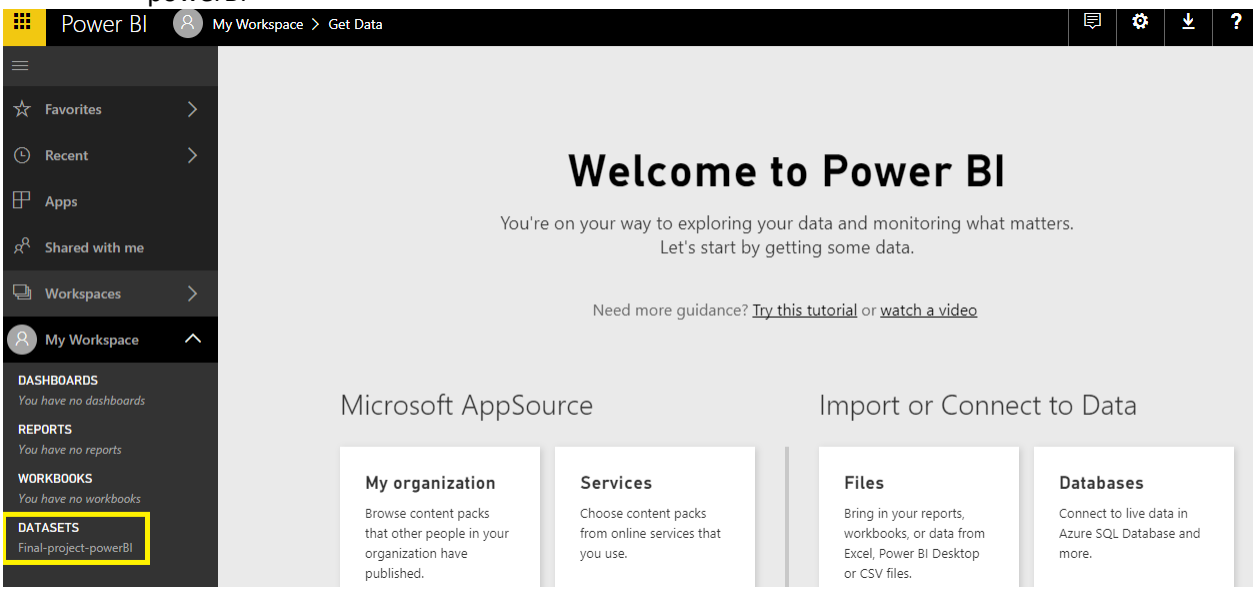
- Now test the data and the output generated is shown below. It is giving the real time fraudulent calls identified. We will have to make sure the Telco app is running.



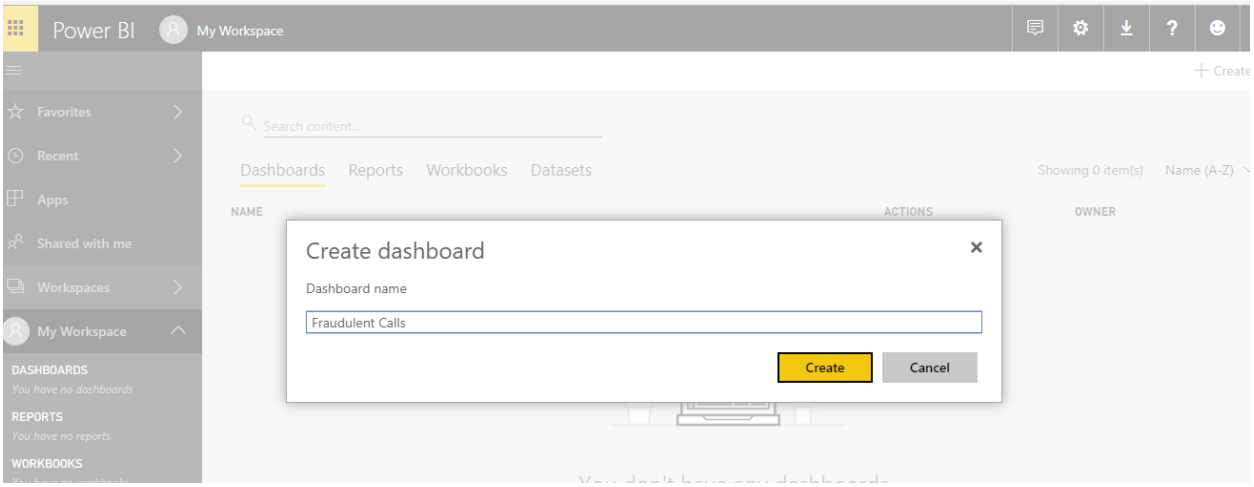
- We will start the job and it will start looking for fraud call in the incoming data.



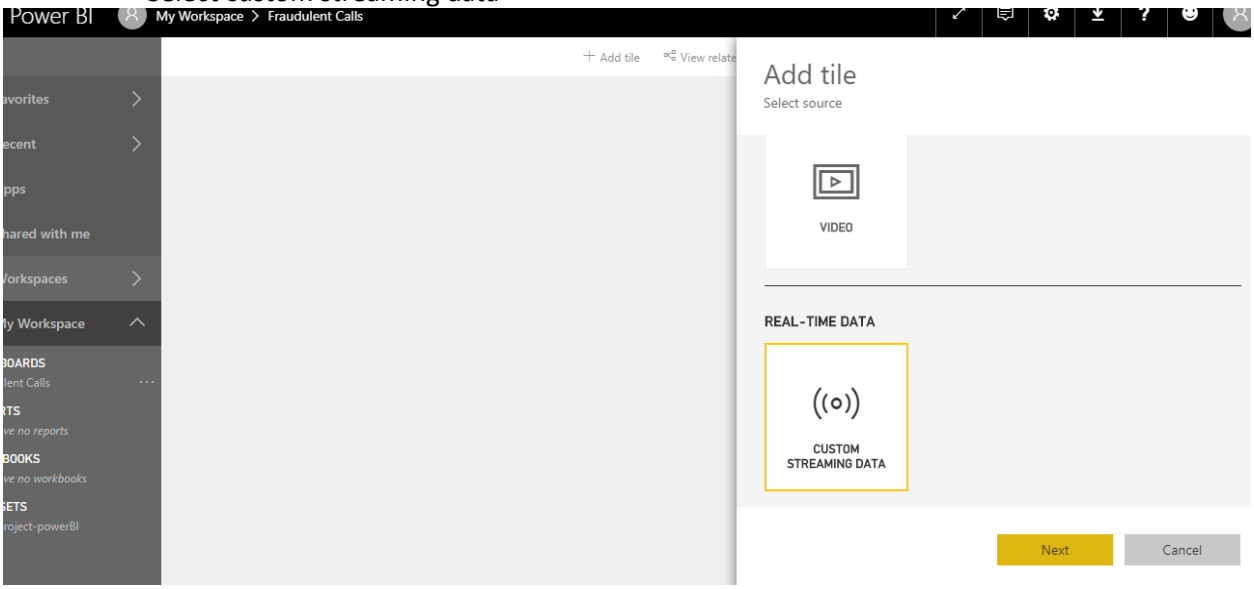
- To put the data for analytics in Power BI, we will go to the Power BI dashboard and clock on My Workspace, we can see the data set we earlier created for the job called Final-Project-powerBI



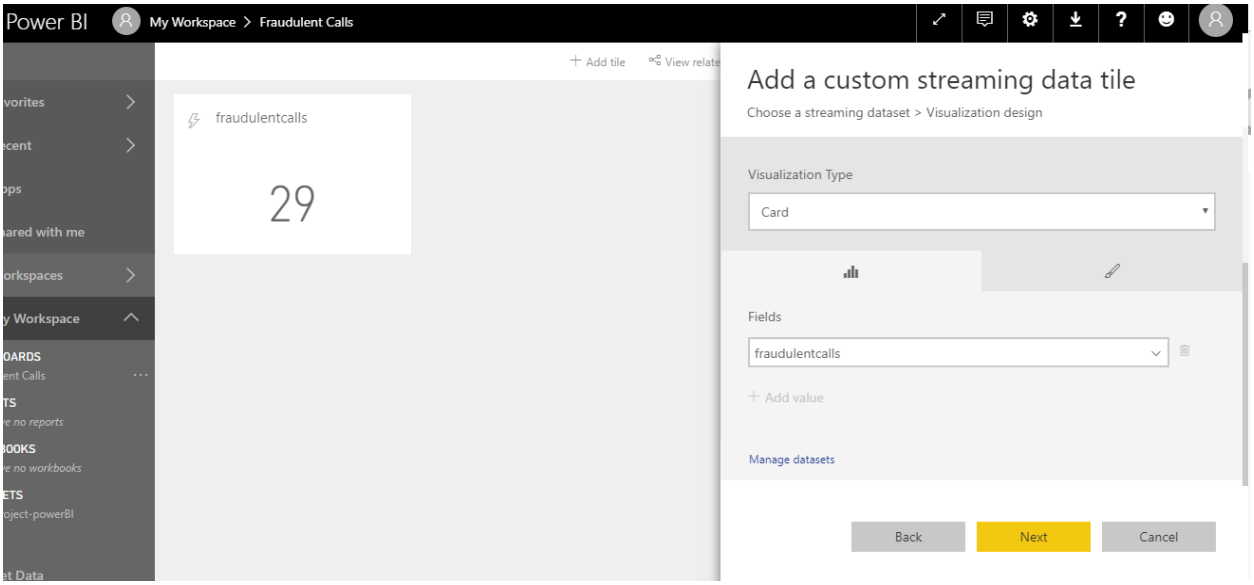
- Create a new dashboard with the name Fraudulent calls

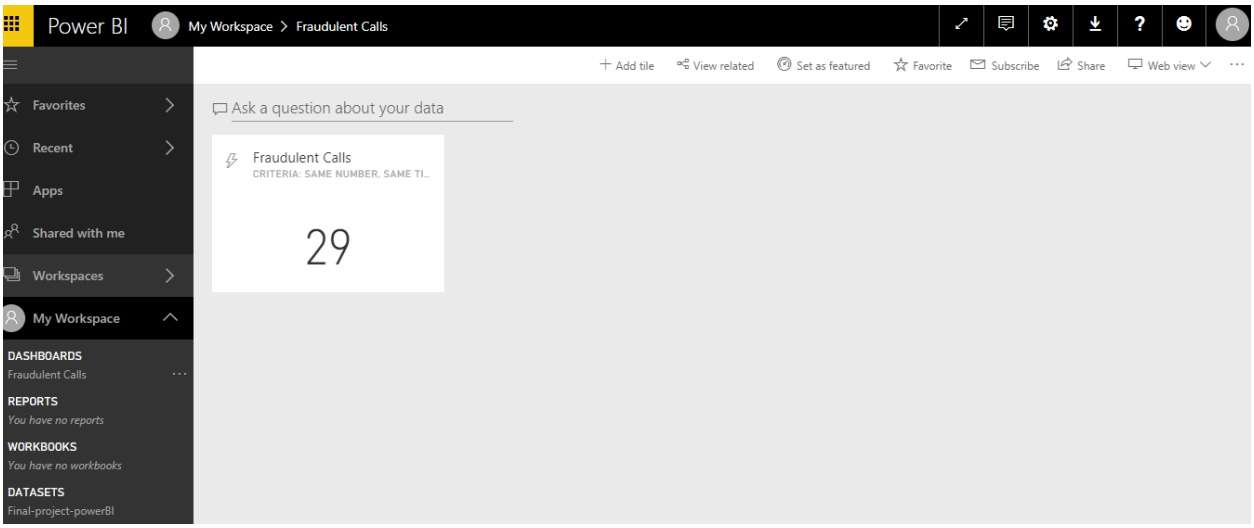


- Select custom streaming data

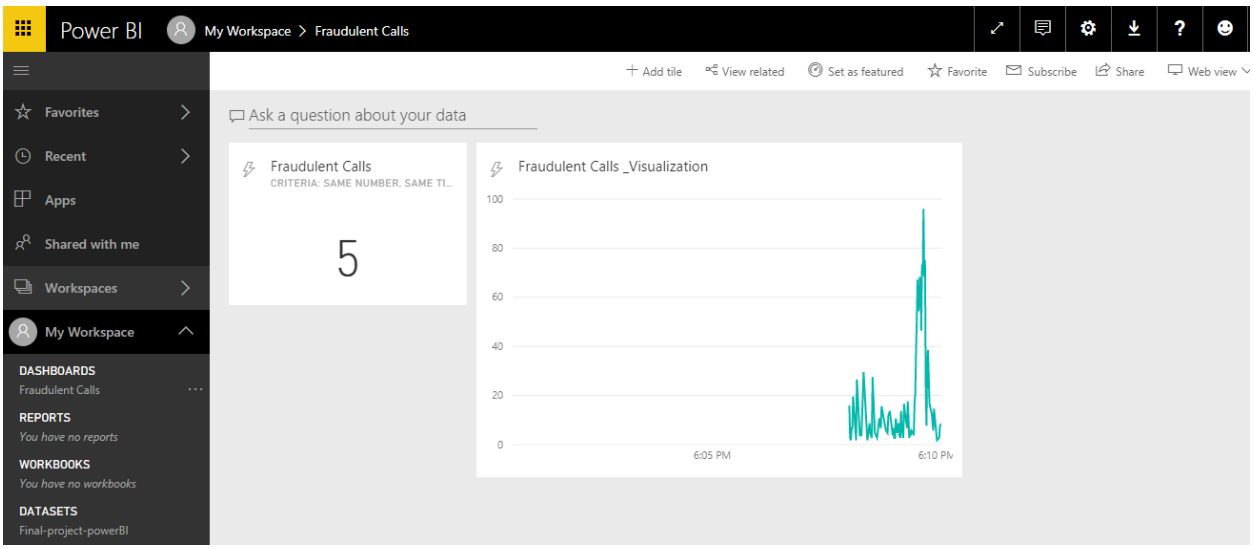


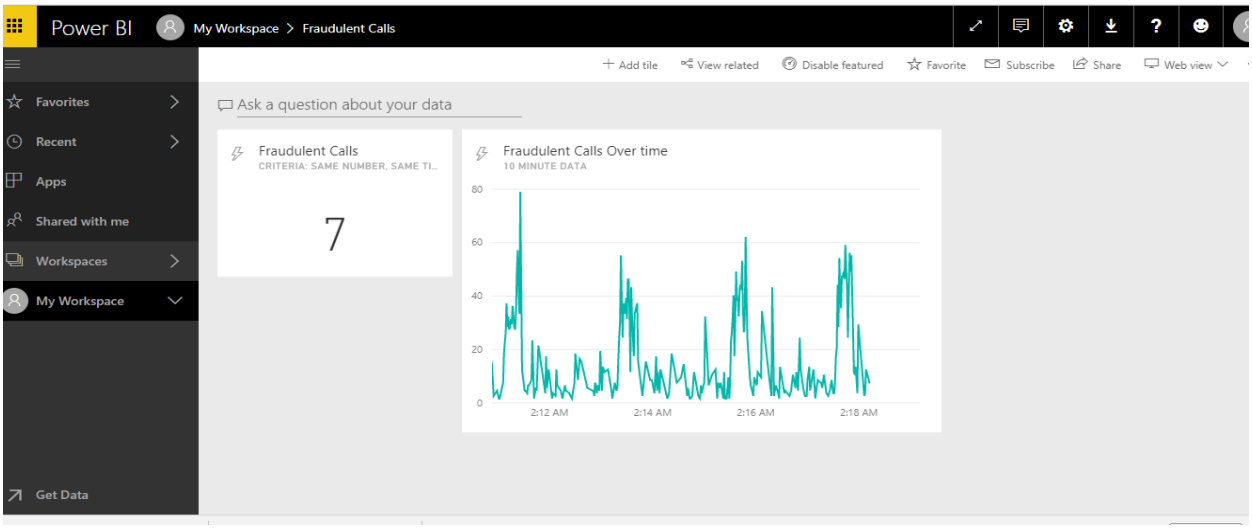
- Here we can see the number of fraudulent calls in the form of a card.





- Also, we can create a line chart showing the graph for real-time 10 minutes data about fraudulent calls.





As recommended by Professor, I again changed the output sink to store the generated output (json file for fraudulent call details)in a storage account on Azure called Blob storage, which is Azure's equivalent of S3 bucket, so that it will be accessible later for analysis.

Summary

The task was to examine a real-time stream of incoming phone calls data for fraudulent calls with the help of Azure Stream Analytics job. The criteria used for identifying the fraud calls was calls received from same number at the same time from different locations. Then for further analysis, either putting that data in Blob storage which is AWS S3 equivalent or using Power BI business Intelligence tool offered by Microsoft Azure to visualize the trends and patterns. As we log into the azure account, we can see the running stream analytics job and generated output being stored on azure cloud storage account called "Blob storage".

References: https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction
https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-power-bi-dashboard
https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction