My name is venkata and don't ask me my full name its pretty long haha.

i am working as data engineer in Goodyear tire and rubber company here in silicon valley. It is 3rd largest tire manufacturer in the united states.

my undergrad in 2014 and masters in 2017 both in computers background.

my past experience is in AWS Engineer,cloud engineer and

Docker vs Kubernetes:

Docker

* I used Docker especially with ECS and we have created recently we moved some of our ec2 Autoscale groups to ECS there I worked with app dev teams I Created a Docker file where you take an image if it’s a java application we will take java based image or on top of that we add layers in the Docker file and finally the run command would specify what runs when Docker file gets converted to a container.
* Then we push the Docker file on to ECR which is the registry at AWS then based on that image that particular tag for the image that was pushed on to ECR.
* We created a task and wrapped that task into a service and the auto scaling properties at the container level are specified at the service and submit the service to ECR and which is the service that does the Orchestration for the container and given the ECS Cluster it provisions the containers so the resources the ECS cluster are optimally utilized.
* The other one that we are looking at right now though it’s not alarmed here trying to experiment with Kubernetes to see what advantages can we get with kubernetes over ECS but the norm here is if there is a managed service available from amazon the first preference is to go with it but given the amount of buzz that kubernetes has out there is just curious about what it has to offer over ECS.
* ECS also provides auto scaling group capabilities. (consistently spread load)

P2:

* I have used ECS professionally that is similar to Autoscale groups but instead of VM s we can containers
* We create a task and the task tells how to provision and manage the ECS instances and like the image is that we create using Ansible.
* On the ECS side we instead create the container images and the container image is typically a text file known as a Docker File and in that Docker file what we do is we take a base docker image and then we create layers on top that.
* I will say take this base image, then copy these files from my local system on to that image and then run this command on that image, then install this software on that image, and all those are specified in the docker file itself and that docker file is what is equivalent to AMI image Id in case of Virtualization.
* So along with the tasks and the docker file the ECS take care of the orchestration for all.

Kubernetes:

* It is like an open source ECS for us.
* So, whatever ECS is does in amazon can be done by kubernetes and in fact kubernetes is probably much more robust than that it is open source so we have to use kubernetes on our own.

Functionalities missing in ECS when comparing ECS with kubernetes:

* It has lot of orchestration capabilities that are well tied to docker even though ECS is docker behind the scenes, they don’t directly expose it as Docker, so there is always catch up to do ECS for bringing the latest docker capabilities.
* But kubernetes is you know something that works well directly with docker.
* So, if you need docker containerization you can do that and you can use kubernetes or apache mesas one of the frameworks to do the orchestration but ECS is strictly associated with amazon containers itself which are of course Dockers behind the scenes but definitely you can’t run them as your native docker containers we have to go through ECS but with kubernetes we can manage any kind of Docker Containerization.

Lift and shift:

* those databases would be relational. so we use DMS and let it migrate it to RDS-6.
* scalability and performance is taken care by RDS
* no, if not RDS  
  We are in the process of migrating from an on-premises Oracle RAC environment to Amazon RDS for PostgreSQL. Our first step is migrating Oracle on-premises to RDS for Oracle. We found that we had two different sets of requirements that DMS helped fulfill. The first requirement is to replicate data off of Oracle to PostgreSQL. DMS has provided us a great solution to address this requirement. In addition, we need to keep certain code tables in sync across multiple database instances on different database platforms. DMS has provided us the technical solution to fulfill that requirement as well. We are currently moving other workloads to Amazon. We are now looking at replacing a replication product we are using with DMS.

<https://aws.amazon.com/dms/>

Devops-CI/CD:

* what extend devops piepeline look like?
* A typical CI/CD pipeline is linear and one-directional. It is composed of a few stages, or phases. The first phase (CI, or [Continuous Integration](https://www.martinfowler.com/articles/continuousIntegration.html)) takes a commit from the mainline (usually referred to as trunk or master, depending on your version control system), and runs a few steps to verify that commit.
* The actual list of steps depends highly on the language and platform used, but typically consists of (at least) checkout, compilation and running unit tests. Some pipelines also add steps to perform code analysis and run additional tests (such as integration tests).
* 
* The pipeline should give early feedback and be as fast / short as possible. I typically try to keep my pipelines finish within 15-20 minutes. To achieve this, it makes sense to parallelize steps as much as possible, so that test failures or other problems can be reported quickly.

#### Building and publishing artifacts

* When all these steps are successful, a unique artifact is built, packaged, and published to a repository. This can be a JAR, .tar.gz file, a container image, or whatever is applicable to the chosen language and platform.
* Some pipelines deploy the generated artifact to a test environment, to run additional checks. I
* Next, the artifact is deployed to staging/acceptance (or whatever you want to call it, the environment should be equivalent to production) and verified.

#### CD?

* Now this is where *Continuous Delivery* and *Continuous Deployment* will start to diverge. If you want to do the former, there will be a manual gate, usually in the form of a button or similar. Pressing the button is required to “promote” an artifact to production.
* When and what to promote is a choice. Promotion typically happens when a build is considered to be “good enough”. QA has run their tests, the Product Owner has signed off, etc. This means that not all artifacts make it to production!
* *Continuous Deployment* gets rid of the manual gate and fully relies on automatic verification of the acceptance environment to determine whether the pipeline can continue on to production. Production, in turn, is verified in the same way. I wrote a little bit about the difference between the two CD’s here: [CD: Continuous Delivery or Continuous Deployment?](https://www.michielrook.nl/2016/07/continuous-delivery-continuous-deployment/)

# Monitoring Tools

AWS provides various tools that you can use to monitor AWS IoT. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

## Automated Monitoring Tools

You can use the following automated monitoring tools to watch AWS IoT and report when something is wrong:

* **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods. For more information, see [Monitoring with Amazon CloudWatch](https://docs.aws.amazon.com/iot/latest/developerguide/monitoring-cloudwatch.html).
* **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see [Monitoring Log Files](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatchLogs.html) in the *Amazon CloudWatch User Guide*.
* **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action. For more information, see [What is Amazon CloudWatch Events](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatchEvents.html) in the *Amazon CloudWatch User Guide*.
* **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see [Working with CloudTrail Log Files](http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-working-with-log-files.html) in the *AWS CloudTrail User Guide*.

## Manual Monitoring Tools

Another important part of monitoring AWS IoT involves manually monitoring those items that the CloudWatch alarms don't cover. The AWS IoT, CloudWatch, and other AWS console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files on AWS IoT.

* AWS IoT dashboard shows:
  + CA certificates
  + Certificates
  + Polices
  + Rules
  + Things
* CloudWatch home page shows:
  + Current alarms and status
  + Graphs of alarms and resources
  + Service health status
* In addition, you can use CloudWatch to do the following:
  + Create [customized dashboards](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/CloudWatch_Dashboards.html) to monitor the services you care about
  + Graph metric data to troubleshoot issues and discover trends
  + Search and browse all your AWS resource metrics
  + Create and edit alarms to be notified of problems

AMAZON MARKET PLACE:

* Software is made available to a user through an [Amazon Machine Image](https://searchaws.techtarget.com/definition/Amazon-Machine-Image-AMI) format, which provides the user with all the necessary information to run the software with existing programs in his or her environment.

ticketing system:

* OTRS is an Open source Ticket Request System (also well known as trouble ticket system) with many features to manage customer telephone calls and e-mails. The system is built to allow your support, sales, pre-sales, billing, internal IT, helpdesk, etc., department to react quickly to inbound inquiries.
* Open Source Software is free but your time isn't and JumpBox is here to help with commercial solutions to simplify the deployment and management of Open Source software on Amazon EC2. This JumpBox is one of a library of over fifty pre-configured, regularly updated AMIs with a consistent deployment and management experience.

Cloud Security:

* Network firewalls built into [Amazon VPC](https://aws.amazon.com/vpc/), and web application firewall capabilities in AWS WAF (web application firewall) let you create private networks, and control access to your instances and applications
* NACLS-whitelist/blacklist ip address.
* Secuirty groups by allowing only wat is required.

DDOS: Distributed Denial of Service

<https://d1.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf>

* Services that are available within AWS Regions, like Elastic Load Balancing and Amazon Elastic Compute Cloud (EC2), allow you to build DDoS resiliency and scale to handle unexpected volumes of traffic within a given region. Services that are available in AWS edge locations, like Amazon CloudFront, AWS WAF, Amazon Route 53, and Amazon API Gateway, allow you to take advantage of a global network of edge locations that can provide your application with greater fault tolerance and increased scale for managing larger volumes of traffic. The benefits of using each these services to build resiliency against infrastructure layer and application layer DDoS attacks are discussed in the following sections

Cloud Encryption:

* Data encryption capabilities available in AWS storage and database services, such as [EBS](https://aws.amazon.com/ebs/), [S3](https://aws.amazon.com/s3/), [Glacier](https://aws.amazon.com/glacier1/), [Oracle RDS](https://aws.amazon.com/rds/oracle/), [SQL Server RDS](https://aws.amazon.com/rds/sqlserver/), and [Redshift](https://aws.amazon.com/redshift/)
* Flexible key management options, including [AWS Key Management Service](https://aws.amazon.com/kms/), allowing you to choose whether to have AWS manage the encryption keys or enable you to keep complete control over your keys
* Encrypted message queues for the transmission of sensitive data using server-side encryption (SSE) for [Amazon SQS](https://aws.amazon.com/sqs/)
* Dedicated, hardware-based cryptographic key storage using AWS CloudHSM, allowing you to satisfy compliance requirements

Enterprise Connectivity:

# we might have it but unfortunately i never got chance to work on it.

# AWS Direct Connect

* AWS Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS.
* Using AWS Direct Connect, you can establish private connectivity between AWS and your datacenter, office, or colocation environment, which in many cases can reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections.

## Service Highlights

### Reduces Your Bandwidth Costs

* If you have bandwidth-heavy workloads that you wish to run in AWS, AWS Direct Connect reduces your network costs into and out of AWS in two ways. First, by transferring data to and from AWS directly, you can reduce your bandwidth commitment to your Internet service provider. Second, all data transferred over your dedicated connection is charged at the reduced AWS Direct Connect data transfer rate rather than Internet data transfer rates.

### Consistent Network Performance

* Network latency over the Internet can vary given that the Internet is constantly changing how data gets from point A to B. With AWS Direct Connect, you choose the data that utilizes the dedicated connection and how that data is routed which can provide a more consistent network experience over Internet-based connections.

### Compatible with all AWS Services

* AWS Direct Connect is a network service, and works with all AWS services that are accessible over the Internet, such as Amazon Simple Storage Service (Amazon S3), Elastic Compute Cloud (Amazon EC2), and Amazon Virtual Private Cloud (Amazon VPC).

### Private Connectivity to your Amazon VPC

* You can use AWS Direct Connect to establish a private virtual interface from your on-premise network directly to your Amazon VPC, providing you with a private, high bandwidth network connection between your network and your VPC. With multiple virtual interfaces, you can even establish private connectivity to multiple VPCs while maintaining network isolation.

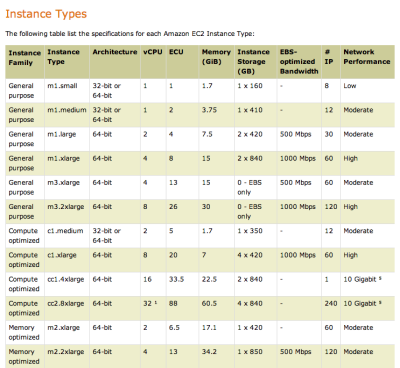
### Elastic

* AWS Direct Connect makes it easy to scale your connection to meet your needs. AWS Direct Connect provides 1 Gbps and 10 Gbps connections, and you can easily provision multiple connections if you need more capacity. You can also use AWS Direct Connect instead of establishing a VPN connection over the Internet to your Amazon VPC, avoiding the need to utilize VPN hardware that frequently can’t support data transfer rates above 4 Gbps.

### Simple

* You can sign up for AWS Direct Connect service quickly and easily using the [AWS Management Console](https://console.aws.amazon.com/directconnect/home). The console provides a single view to efficiently manage all your connections and virtual interfaces. You can also download customized router templates for your networking equipment after configuring one or more virtual interfaces.

Ec2 instances:



Components of microservices:

## Properly scoped functionality: The biggest design issue with monolithic application architectures is that there’s so much code in them that implements widely differing functionality. To make any change to a monolithic app, you must coordinate across different groups in order to ensure that everyone’s code continues operating properly. As a result, developers often spend more time on integration and testing than on delivering new application capability

## Presenting an API:Once you break up a single application into multiple cooperating services, how should the services talk to one another? Typically, this is done with REST web services API calls, although you can use other transport mechanisms as well.Presenting an API to calling services in some way represents the old challenge of integration. For an overall application to run properly, each of the individual services must be able to reliably send and receive data, and testing that APIs operate properly is necessary to ensure that everything hangs together.

## 3. Traffic management:Once the API is up and one service can call another, everything’s OK, right? Well, no, actually.In the real world of operations applications, a service may run slowly, and calls to it to take a long time. Or a service can be overwhelmed with calls and lack the processing power needed to respond quickly enough. Even worse, a service might simply stop running due to a software or hardware crash. And sometimes a client is issuing too many calls for the lower-level service to respond quickly enough.Addressing this too-heavy traffic situation requires management. There must be a way for calling and called services to communicate status and coordinate traffic loads.

## 4. Data offloading:The vagaries and erratic traffic of microservice applications mean that individual services come and go. Adding to the constant service instance churn: the reality that the underlying infrastructure also is unreliable. Virtual machines crash, fail to respond, or go into high-load status while not performing any useful work (thereby requiring hard termination). Nevertheless, while individual services instances are transient, the overall service must be available and continue operating so that users will keep obtaining results from the application.

## 5. Monitoring:Decomposition of a monolithic application, along with insertion of offloaded data layer and caching to increase performance, inevitably means a more complex application topology — a lot more complex.

API GATEWAY:

* Microservices usually expose REST APIs for use in front ends, third-party applications, and other microservices. A best practice is to [manage these APIs with an API gateway](https://aws.amazon.com/api-gateway/). This provides a unique entry point for all of your APIs and also eliminates the need to implement API-specific code for things like security, caching, throttling, and monitoring for each of your microservices. You can implement this pattern in a few minutes using Amazon API Gateway. Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.
* https://aws.amazon.com/blogs/compute/using-amazon-api-gateway-with-microservices-deployed-on-amazon-ecs/

IOT:

Goodyear-3rd largest retail in usa-manual data collecction-predictions-PSI-Streamingdata

* AWS IoT Device Management makes it easy to securely onboard, organize, monitor, and remotely manage IoT devices at scale.
* IoT Device Management lets you register your devices individually or in bulk, and manage permissions so that devices remain secure.
* Then, you use the IoT Device Management console to organize your devices into groups, monitor and troubleshoot device functionality, and send remote updates to your devices. AWS IoT Device Management allows you to scale your device fleets and reduce the cost and effort of managing large IoT device deployments.
* <https://aws.amazon.com/iot-device-management/>

What you have done on AWS Roles?

* I can say like lot of my current experience is more on the Devops kind of roles
* Basically, I have performed various roles on Aws like I have done devops kind of roles as well as what diff companies call as a developer role also analyst roles on AWS.
* I concentrate more on setting up the infrastructure the automation of infrastructure based on Cloud Formation and also work with databases and migration of apps in the cloud as well as spinning up infrastructure and automation of infrastructure in the cloud.
* My experience is more towards automating the lot of the infrastructure creation and also helping development teams use the right managed services that are the building the solution. For example, if its

Streaming middle ware we use Kinesis, and s3 as work back bone for data storage, Dynamo Db as our transactional Database store and Redshift as our data ware house and also for big data analysis jobs we use EMR and spark on top of EMR

* So that is where I have most experience in and revolves around lot of these AWS services and all these in building native solutions in the cloud.

What exactly you have migrated from on premises to cloud?

* So in the current role the part of the solution for the data lake pretty existed on premises, but lot of the solution was built on top of Db2 as the database and also was utilizing custom built solution as a messaging middle ware. Ok.
* So what we have to do is a lift and shift of the solution , we have to take the real time middleware as is and it was built on top of scala and akka, but we couldn’t take the db2 into the cloud bcz its too costly and resource intensive to migrate db2 into the cloud.
* So what we have to do is, though we did a lift and shift on the middleware custom made middle ware for real time ingestion, but for database we have to extract the data from db2 using native db2 utility migrate that onto the extract it to s3 and then use the datapipelines to read that data from s3 and load it into an RDS instance.
* So that is what we did as part of lift and shift of the application

What kind of AWS resources did you use while migrating from on premises to cloud?Did you use any migration service?

* The Db migration service that comes with AWS supports all DBs except Db2, since we had Db2 on premises their was no way for us to use any of the migration service from Amazon bcz AWS DNS does not support Db2.
* So we had to manually extract the data load it into S3 and start using datapipelines

So what was the approximate data that you moved from on premises t cloud?

* So it was more the machine and the economic data, so the way the system is architected on premises is they have the data sharded in Db2 as (it has multiple databases they call them as shards) and few hundreds of machines are allocated to each database and there are around 50 of such dbs here and each db can go upto anywhere between 100 – 200 GB.
* So we have to migrate each one of the database as a single migration

How long did it took for migration?

* So the extract itself was really very fast bcz it was native extract utility on the mainframe for Db2 and getting the data to s3 was one time upload process, but once the data was their migrating the data to RDS took anywhere btwn 4 hrs - 6 hrs.

How did you manage your EMR clusters and what did you do in terms of like Jobs(tasks)?

* So from my roles perspective I have used cloud formation to enable the provision the EMR stack, but I know that the developers actually use the EMR stack by submitting spark jobs, that they would do some analysis with data on using this sparks jobs that run on top of the EMR cluster.
* But I was not personally involved in the development of those spark jobs.

Can you elaborate little bit on spark jobs, EMR and batch jobs?

* Ya what I can say is sometimes the data flows through multiple stages like the ingest, model, enhance and transform, at each of these phases that the a copy of the data is actually stored in S3.
* So in some cases what happens is we have to test the data or test the replay of this data mechanism on this data.
* So actually they would run some EMR jobs that would take the data from the previous phase and then apply the logic as it is the input for the next phase to validate the some of the replay mechanisms that would have to be run,if their was some loss in the data.
* So they use to use some EMR jobs to do that data processing from one stage of the datalake to the other stage

Were you involved in the EMR jobs?

* I was not particularly involved in the development of the EMR jobs, as I told you I was involved in the provisioning of the clusters and provisioning of all the AWS resources.
* The stacks that were to be provisioned to enable the development teams to develop and run their Spark jobs

Do you know what other applications can we launch using EMR other than spark?

* Ya EMR basically is a Hadoop as a service, so any kind of Hadoop jobs can be run on an EMR cluster.

You said that you designed and developed data ingestive , transformation phases of data lake? Can you explain me how you were involved in that?

* Like I said data here is both machine and economic data that is ingested into the data lake eco system here.
* So as part of the ingestion phase the couple of things that I was involved in designing and developing was , their was a requirement for the copy of the data as it is ingested into the system to be stored, so we know the source of truth of how exactly the data made it into the system.
* So what I did was we took the data ingestion and through kinesis firehose we send the data to a kinesis firehose and bundled this data based on an interval of 5 min.
* The chunk of the data as it arrived was put into the S3 . so this was the original data as it is recieved or ingested by the system, so that is the data ingestion phase.
* Once the data is ingested the stream of the data actually has to make it to the next phase after it is successfully put it into the s3.
* The next phase was actually a model phase, the way we handled the data flow between any two phases was by using kinesis as a streaming middle ware.
* So the the data from the ingest phase was put into the kinesis stream and the next phase has to pick it up from their and that will also allow us to control the back pressure if one of the downstream phases has a slow processing capacity than the one before that, so that is the ingestion phase.
* So transformation is one of the later phases in the life cycle after ingestion, model and enhance so that, and transformation phase is were the business logic is applied , so that is were the data has to be actually stored in a dynamo db .
* So their we used the dynamo db and u know and designed some dynamo db document models and that would allow also the provisioning of the dynamo db and capacity planning of the dynamo db based on how much capacity we anticipated the system to receive.

Were I was involved ?

* So some of the things where I was instrumentally involved was data transformation logic itself is a java based microservice.
* So I was instrumentally designing and developing their release pipeline where these java based micrservices which are actually maven based project ok.
* So we take that maven based microservices and design CI / CD pipeline where once the code is checked into the git hub it goes through the Jenkins job where I wrote the jekins jobs using the Jenkins DSL and then using the Jenkins 2.0 pipeline feature , the code that is check into github will trigger the Jenkins job and that Jenkins job will take the code compile it and run the unit test .
* Then generate the artifact the jar file and the upload the jar file into Artifactory , so that is the build phase of the job once the build phase is done their will be deploy phase where the artifact that is uploaded into Artifactory is actullay retrievd of the Artifactory again and then it is deployed on to the auto scale group .
* That is the auto scale group , that the cluster of the transformation phase that the deploy job will take care of deploying it to the auto scale group and provisioning of the infrastructure for this auto scale group is defined as the separate stack called the transformation stack.
* Because each of the phases have their own stcak name defined so I was also involved in the development of this stack , that will go ahead and provision the infrastructure that is needed for the autoscaling group of the transformation stack.

Two steps :

Step execution mode

What were the problems that you faced while using elastic search ?

* While using elastic search stack …. The major problem was not itself the elastic search, lot of developers did nt know how to move from what they are used to the appl logs were always log to a log file that s the kind of mentality most of the developers use to.
* So the elastic search with kibana the learning curve between looking into a log for your

Can you explain Structure of Cloud Formation template?

* Cloud formation is infrastructure as a code and I would rather say CF is Infrastructure as configuration.
* We go ahead and specify the resources in cloud as a json or yaml file and for each resource we give the name of the resource and type of the resource is what type of resource it is whether it is an Ec2 instance or Rds instance or Aurora database or s3 bucket etc. and then based on type of the resource we specify parameters or properties of the resource based on which the resource is provisioned in Aws.
* So we describe the infrastructure that we need as various resources and then we specify the parameters that are required for each of those resources and we can also specify what we need to have in the output section when this particular template is provided a corresponding stack is created in aws and we specify the output section what the output should be that we would like to see when the stack is created and to make the stacks dynamic we can pass in parameters or specify parameters in the parameters section of the cloud formation template.

If I want to create a VM using cloud formation template can you walk me through what are the configurations required?

* When it comes to aws VM’s are called EC2 instances that is the virtualization the compute virtualization at aws.
* So typically, when we create ec2 instances even though we can create ec2 in isolation typically what we do is we create what is known as auto scale group we give a name to it and specify that desired capacity which is the number of instances that reside in that auto scale group.
* In this case we can give the desired capacity as 2 we can give minimum and maximum capacity and then we can also specify the scale up and scale down parameters based on which the capacity of auto scale group will either increase or decrease and for that we also specify the alarms definition in the cloud formation template itself.
* For example, one that says if cpu capacity of the auto scale group goes beyond 90% for at least 5 minutes then increase the capacity by 1 instance likewise decrease instance by 1 if cpu capacity is below 40% that is all these are defined as parameters or properties of auto scale group in the CF template and that CF template is executed a corresponding stack is created with that Autoscale group and the number of instances in that auto scale group will be equal to desired capacity of the Autoscale group.

What is called the Virtual Network(VPC)?

* Virtual private cloud is the virtual network in aws.
* First thing is we need to do is create a VPC in the cloud and give a CIDR range for that vpc and then create subnets inside vpc and in those subnets, is where we create the actual resources like EC2 instances.
* The only limitation is the CIDR range. So, we have IP space of 1024 addresses inside vpc then we can only create subnets that can go up to 1024 IP addresses.
* One limitation that is there with the VPC CIDR block is we cannot go beyond 65000 (65536) distinct IP addresses for the CIDR range. Any number that is less than 65000 should be ok means /16.

What is the use of Gateway?

* We have different gateways in the cloud the main gateway that is typically used is what is known as internet gateway.
* It is attached to VPC so that internet traffic can flow in and out from vpc to internet.
* There the constraint is one vpc can only have 1 internet gateway attached and 1 internet gateway can only be associated with 1 vpc.

10 applications in 1 vpc and we want to create DNS for each and expose them to customer?

Within vpc or outside vpc?

In vpc - 5 subnets and 5 have 5 apps and exposing to public customer and I have to create DNS for all 5 apps so if I want to configure through gateway what should I do?

* If we want to expose application running in 5 different subnets as 5 diff applications to outside world then what we need to do is first we need to create a route table to route the traffic from through the internet gateway associated with the vpc.
* Then that will allow packets to flow from and to the internet through this application.
* We have to create a route 53 entry and we will be able to share a DNS name to the end users that they can access these applications through.

Suppose I have 2 applications in 2 different subnets……then how can I work this situation

* First thing we need is our route table that routes the traffic through the internet gateway and then

We need to create DNS entry in route53 so we can provide friendly url that the customers can use to communicate with the applications from open internet.

CI/CD

* What we have used for CI/CD we used AWS products and also Jenkins.
* I will explain what AWS products we used for CI/CD. So, when application code gets checked into GitHub that GitHub through web hooks go and trigger a code pipeline.
* As part of code pipeline the first stage there are different stages in code pipeline.
* The first stage of code pipeline it will go ahead and check out the code from GitHub that is what is known as the source stage and in second stage what is known as build stage and this is where we use the code build service at AWS and that it will go ahead and build the code like if it is a java app it will create a jar file or war file and if it’s a node.js application then it will go ahead and create the corresponding package etc.
* So, depending on the technology we are using the build will use the corresponding technology like maven in case of java or npm in case of node apps or pip in case of python and then it will create a deployment package.
* Then this deployment package can go into a repo service like Artifactory or in some cases we put that package in S3 bucket.
* Once that build phase is done then comes the deployment phase.
* This is where the other service at AWS known as code deploy comes into picture and it will grab that package either from Artifactory or s3 bucket and then it will deploy package onto Autoscale group (that has APPSPEC.yaml that is a configuration file that says to deploy the package what it should do. For example, the APPSPEC.yaml file we can say copy this war file into the tomcat apps folder then all the ec2 instances in that Autoscale group will have tomcat installed and this war file will be copied on to a particular directory and each of this ec2 instances and then tomcat services can be restarted as another bash command that we can specify in the APPSPEC.yaml deployment configuration file) for example and then that is the deploy phase.
* This whole thing is defined as a pipeline in the code pipeline service that is another AWS service and this code pipeline service is hooked to a GitHub repository or some cases some people may also use code commit but most of the projects I worked on I use GitHub.

SAML 2.0

* This particular client actually uses Octa as their identity provider but even before Octa came into picture here they actually use ADFS. So, the users actually login using single sign on based on ADFS and there from there they login into central account.
* There are no users provisioned, based on the role definition given as an attribute in the active directory. using that role, they login into a central account and from there their role is propagated into corresponding department account so from there administration VPC that is where ADFS integration is built in and beyond that the users go into corresponding department account in the cloud.

Amazon KMS (key management)

* One pattern that we use here is if there are any property files that are needed during the provisioning process the OPS team signs that using KMS key and Jenkins job, the role that Jenkins job assumes will have the IAM authority on that key. So, it will actually decrypt the key after getting it from S3 before it injects the key onto VM in which the property file is required so we use KMS for one of the purposes.

Canary deployment or blue green deployment:

* We call canary deployment as blue/green deployment.
* If we have to deploy a newer version of the application then we just keep running the existing version as it is and we create a newer version of the application infrastructure and deploy the newer version of the application on to the infrastructure and then verify that the new application on new infrastructure is working fine and then route the traffic c onto that newer instance only after all the tests are successful.

CI/CD vs Jenkins

* Jenkins is something that we have to maintain that we have to create infrastructure to deploy Jenkins and maintain Jenkins.
* Jenkins pipelines are defined in what is known as Jenkins 2 pipeline dsl that is groovy dsl.
* So, the pipeline definition itself is written in groovy and that is provided to the Jenkins to create a pipeline job.
* But otherwise the concepts are similar. The Jenkins pipeline also has multiple stages where we will have the source stage, build stage and deploy stage just like what code pipeline has.
* But in case of Jenkins we have to create the infr astructure to run the Jenkins master and multiple slaves that are connected to the master we have to maintain the plugins and we have to maintain availability of the Jenkins and all that.
* But in case of code pipelines we don’t have to maintain any infrastructure or build any infrastructure because these services are offered by managed services that are offered by amazon.

AWS Services/components Used:

* I have used quite a few aws services.
* If I have to name some of the prominent one’s on the database side it would be dynamo and aurora db.
* I have also used SNS and SQS and cloud watch log for notifications and I have used CF for automation of infrastructure provisioning and like I said I also used kinesis for streaming based solutions and I have also used elastic search for creating elk stack.
* IAM for security and s3 for object storage and lambda for Serverless computing models.
* Those are some of the major at high level services that I have used.

What did you do in Elastic Search:

* The elastic search is basically document database that also happens to create an index for all the documents that it stores and these docs are stored as json.
* One thing that we did using elastic search is to create an elk stack.
* So, we created an elastic search cluster and we created 2 indexes in the cluster out of which 1 index cluster was for application logs and other is for infrastructure logs.
* For the index that was created for application logs what we did was we changed log configuration file in the applications to redirect the logs to elastic search instead of writing it to a log file on the local machine.
* So, all the apps now send their log files to elastic search, at the same time the infrastructure logs, all the logs that come from ec2 instances that contain the events that generated by infrastructure itself those were sent to cloud watch and for cloud watch we redirected those events into other index file that we created for the infrastructure logs.
* Now since elastic search service itself comes with Kibana we were able to define some queries in Kibana so the developers can go and look at the applications log files across different components combine them to look at the exact logs that they want and also same with the infrastructure log files.
* Elastic search can only take json as input.
* Data ingestion is done in json not just for logs we can ingest any data into elastic search but the document format is JSON.
* Data format can be anything but the Document format inside your index is Json.

Pushing the logs to elastic search Instead of writing logs to disk how did you do that?

* Yes, that’s true, the appenders that we have in the log typically are file appenders which means the logs are appended to a file that is typically how applications are working today but there are also known as elastic search appenders where we can use something like Kibana which will take these logs from apps and redirect those logs to elastic search that is the ‘K’ in elk stack.

If we do that we have more overhead to the application, right?

* No application is still sending the events.
* Instead of writing the events to a file it is just sending the event to Kibana and in some cases what happens is even the application team is not willing to do that Kibana also has the ability to read the log file generated by the application and then send that to elastic search. In that case there is nothing that has to change in the application but we can do both ways.

Your log is sending to elastic search or log 4 j?

* The logging framework is still the log 4 j but the logs themselves are written or redirected to elastic search and that is where they get stored.

**AWS IAM :**

* IAM is a web service that helps you securely control access to AWS resources for your users.
* Use IAM to control who can use your AWS resources (authentication) and what resources they can use in which ways (authorization).
* In Amazon ECS, IAM can be used to control access at the container instance level using IAM roles, and at the task level using IAM task roles.

ANSIBLE

* How I used Ansible is for infrastructure automation So, what is known as configuration management
* In simple terms, it is used to install software after a virtual machine is created.
* What I have used Ansible for is I create Ansible playbooks and then install software on to ec2 instances and create an image out of that so that image can be used when we provision infrastructure using CF. So, the first step is if there is
* Java app that requires tomcat we write Ansible playbook that will go ahead and install JVM and also Tomcat.
* Then the ec2 instances uses tis Ansible playbook install that software’s on to that ec2 instance and then take the image ID and use that image ID in the CF when we create the actual infrastructure itself
* So typically, Ansible was used to create or install the software that is required remotely.

Chaos Monkey (k-o-ss monkey)

* what it does is it will go ahead and suddenly shutdown some instances that are running in live system.
* So, we will use that to test autoscaling and how autoscaling heals and how fast it heals.
* If particular instance goes down so when we deploy chaos monkey onto a particular aws account or a particular VPC and then we can configure chaos monkey and then say, hey can you go ahead and test the services like that are running on this particular Autoscale groups and suddenly without notice it will go ahead and shoot one or more instances on that group and what that will helps us test is reliability of the system.
* How is the system going to heal when sudden failure of certain components or part of certain components occur and to study the behavior we have used chaos monkey.
* It is an open source software from Netflix.

What is S3 and choose to use it like features and capabilities?

* Most of the storage systems we are used to in the current world are Block based storage systems. Unfortunately, the industries have not realized the potential of an object based storage system. Lot of awareness to data storage system i.e. object storage systems came through S3.
* According to me S3 is typically the backbone for any data storage, especially given that I have worked on projects like Data lake and fleet management that I worked on.
* Especially in the project like data lake S3 is the backbone for data storage, any time you ingest the data into a system, if you want to have a record of your data as you ingested before the software that attempts to read the data, it’s a good practice to have a copy of the data in some durable storage and given that there is no better durable storage than S3.
* Starting from the very first point of your data life cycle, the storage has to start with S3, that’s one great use of S3 i.e. the data that you ingest into the system have a copy as is, in a durable storage system like S3.
* And there can be many users for that where we can reprocess that data or if we need it for auditing etc.
* Then, as the data progresses through its life cycle then at every stage of that life cycle, I typically recommend to have a copy of that data again in S3.
* Some of the Uses of S3 are like:

1. You can have the copy of the data as it goes through its life cycle its important phases of the life cycle different important phases of life cycle.
2. Depending on the system that you are going to build i.e. if it is a web based system then all your static content can go into S3, obviously backed by S3 and served by cloud front which is another use case .
3. It is used for archival, in this i.e. back up based if you are trying to do back up and recovery solution S3 is the go to for your backups and even for archival.
4. And any blob based storage typically in the past we had only one choice i.e. only the relational database, it use to beat that horse to death when it comes to storage right .i.e. whether it is photos in the web application or anything else.

I have seen cases where people store that as a blob field in relational database, with the cloud thank fully, we don’t have to do the sin like storing the photos in a relational databse.

Any time there is a blob kind of storage where you really

Data Lake:

* As part of the data lake we collect the data, or with the real time data ingestion from lot of sensors we get real time data.
* These data goes through multiple phases like : Ingest, Model, enhance and transform before it gets into the transactional database and also from their we create Rest API i.e. to expose the data to the consumers.
* So right from the data ingestion all the way through exposing the data using the API s we are responsible for using all the managed services in the cloud to build the solution and then automate the process of deploying the code across multiple microservices that are the building blocks of this solution.

Do you have any questions for me?

* What is team structure look like and how is the work is it agile/Scrum environment and is it an appropriate time to share those details?
* For automation we used Jenkins so far and recently migrating to code deploy, code build and code pipeline is that same with you?
* The only thing I wanted to know is more about the position?

**Types of Load Balancers**

1. **Application load balancer :** best suited for the load balancing of **http and https traffic** and provides advncd request routing targeted at the **delivery of the modern app architctres**
2. **Network Load balancer :** Network Load Balancer is best suited for load balancing of **TCP traffic** where extreme performance is required
3. **Classic Load balancer :** Classic Load Balancer provides basic load balancing across multiple Amazon EC2 instances and **operates at both the request level and connection level**

**Amazon EC2 Container Registry**

* Amazon ECR is a managed **AWS Docker registry service** that is secure, scalable, and reliable.
* Amazon ECR **supports private Docker repositories with resource-based permissions** using IAM so that specific users or EC2 instances can access repositories and images.
* Developers can use the **Docker CLI** to push, pull, and manage images.

Components of ECR :

1. Registry : An Amazon ECR registry is provided to each AWS account; you can create image repositories in your registry and store images in them.
2. Authorization token : Your Docker client needs to authenticate to Amazon ECR registries as an AWS user before it can push and pull images.
3. Repository: Contains ur docker images
4. Repository policy : Control access to ur repositories and the images within them with repository policies.
5. Image: You can push or pull the docker image to ur repositories. You can use these images locally on your development system, or you can use them in Amazon ECS task definitions

**SSO (Single Sign on) :** As organizations adopt cloud applications in droves, end users end up having to deal with more and more passwords throughout the day just to complete their work.

Users begin suffering from password fatigue and forget their passwords, which leads to password-related help desk calls and reduced productivity.

Or, they start using weak passwords that are easier to remember, but compromise IT security.

For effective user identity management, you need to adopt an efficient and secure approach to managing users’ passwords.

ADSelfService Plus is an enterprise single sign-on (SSO) solution that provides users with seamless, one-click access to cloud applications.

It allows users to enter a username and password just once, and gives them secure access to multiple cloud applications.

Users can now access all their cloud applications with just one identity.

ADSelfService Plus minimizes password security issues, increases productivity, and eases identity management challenges, while also enhancing user experience

**Cloud migration:**  **Cloud migration** is the process of moving data, applications or other business elements from an organization's onsite computers to the **cloud**, or moving them from one **cloud** environment to another.

For example moving the infrastructure of the company from on premises to cloud.

1. they choose to move to the cloud is the agility and speed that they can move with
2. With cloud computing, you are able to spin up thousands of servers in minutes whre as on premisesit takes like 10-18 weeks to spin up the servers being used on premises.
3. AWS cloud provide wide variety of services on cloud

We can see 5 phases in migrating process.

1. **Migration Preparation and Business Planning** : Here we dteremine the rght objectves and begin to get an idea of the types of benefits we can see.

It usually starts with foundational experience and developing a business case for mgrtn.

This require to know the age and the architecture of the existing apps.and their constraints.

1. **Portfolio Discovery and Planning**: we need to understand our IT port folio ad the dependencies btwn the apps, and begin to consider what types of mmigration startergies can be employed to meet our business case objectives.

**3&4**. **Designing, Migrating, and Validating Application**: Now we move the focus from the port folio to individual app level and we design migrate and validate each app. Each app is designed and migrated and validated according to one of the **six common app stratergys i.e**

1. Rehost (“lift and shift”)
2. Replatform (“lift, tinker and shift”)
3. Repurchase (“drop and shop”)
4. Refactor / Re-architect
5. Retire
6. Retain
7. **Operate**: As applications are migrated, you iterate on your new foundation, turn off old systems, and constantly iterate toward a modern operating model.

AWS Migration tools and services:

1. Sever and database migration: In this we have three services available in the aws .

**AWS SMS (SERVER MIGRATION SERVICE)**: this service is an agentless service and which makes it easier and faster for you to migrate thousands of on-premises workloads to aws.

AWS SMS allows u to automate, schedule and maintain the track increamntal replications of live server volumes making it easier for u to coordinate large scale server migrations.

### AWS Database Migration Service: This helps us to migrate the dbtses to aws quickly and securely. The source dbtse remains operational during the migration and it also minimizes the downtime to apps that rely on the database. The aws DMS can migrate ur data from and to the widely used commercial to open-source databases.

### This service also supports homogenous and hetergenous migrations i.e oracle to oracle and oracle to aws aurora respectively.

### This can also be used for continues data replication and high availability. Usually recommended bcoz f the following advantages:

### Simple to use

### Minimal downtime

### Supports most widley used databases

### Low cost

### Fast and easy to set-up

### reliable

### 

### VMWare Cloud on AWS : this service will be a new solution that makes it easy for the customers to use vmware workloads on aws cloud.

### Data Migration: Is seen in some of these ways:

### Aws s3 Transfer acceleration:

### Transfer acceleratn enables fast easy and scure trnasfrs of files over long dstnces betwn the client and the amazn s3 buckt.transfr acclrtn take the max advantage of cloud fronts edge locations. As the data arrived at the edge location data is routed to the aws s3bucket over ana optimized ntwrk path.

### Aws se Transfr acc: makes the public internt tranfrs to s3 more faster. You can maximize ur available bandwidth reagrdlss of the dstnce or varying intrnt weather .Here we cannot even see the spcl clients and ntwrk protocls.

### AwS snowball:

### Snowball is a peta-byte scale data transport solution that uses secure apps to trasnfr large amounts of data into and out of the cloud.

### Using snowball addrss common challenges with large scale data transfrs including high netwrk costs long transfer times and security concerns.

### Transfering data using snow ball is fast, secure, simple and can be as little as one –fifth the cost of high speed internet.

### We don’t need to wrte any code or purchase any hardware to trsnfr ur data.

### We just need to simply create a job in console and a snowball appliance is rrived to you or shipped to u .

### Once it arrives attch it ur local ntwrk download and run the snowball client and copy al the files that u want to trsnfr into the cloud .

### The client thn ncrypt the data and trnsfr the files to the appliance at high speed.

### Once the data is copied and ready to be shipped a EInk shipping label is automatically generated.

### We can track the job status by using AWS SNS , txt, msges or directly through console.

**AWS Snowmobile:** AWS Snowmobile is an Exabyte-scale data transfer service used to move extremely large amounts of data to AWS.

* You can transfer up to 100PB per Snowmobile,
* a 45-foot long ruggedized shipping container,
* pulled by a semi-trailer truck. Snowmobile makes it easy to move massive volumes of data to the cloud, including video libraries, image repositories, or even a complete data center migration.
* Transferring data with Snowmobile is secure, fast and cost effective.
* After an initial assessment, a Snowmobile will be transported to your data center and AWS personnel will configure it for you so it can be accessed as a network storage target
* to connect a removable high-speed network switch from Snowmobile to your local network and you can begin your high-speed data transfer from any number of sources within your data center to the Snowmobile
* After your data is loaded, Snowmobile is driven back to AWS where your data is imported into [Amazon S3](https://aws.amazon.com/s3/) or [Amazon Glacier](https://aws.amazon.com/glacier1/).
* Snowmobile uses multiple layers of security designed to protect your data
* GPS tarckng, alarm monitoring,24/7 vdo survilnce, and an escort security vehicle.
* Data is encrypted wth 256-bit encryptnkeys mnaged by AWS KMS.

**AWS DIRECT CONNECT:**

* AWS Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS
* Using aws direct connect we can establish a private connetvty btwn the aws and ur datacenter office or colocation environment which In many cases can reduce ur ntwrk costs , increase through put and provide more consistent ntwrk exprnce.
* Using industry standard 802.1q VLANs, this dedicated connection can be partitioned into multiple virtual interfaces
* Where we can use the public ip adrss space to make the same connect to access publc resources such as obj in s3. And private resources such as ec2 ruuning wthn the VPC using the private ip addrss space while maintaining separate publc and prvte ntwrk envrnmnts.

### Amazon Kinesis Firehose :

### It is the easiet way to load the data into the aws.

### It can capture automatically load streaming data into s3 and redshift, enabling real time analytics with exstng business intelgnce tools.

### Fully managed servce which automatcly scles to match the throughput of the data and requires no ongoing admsntrtn.

### It can also batch, compress and encrypt the data before loading it.

### Minimizing the storage used at the destination and increainsg security.

Cloud front :

AWS cloud front is acontent delivery netrwrk service that securely delivers data videos apps, and api’s to ur viewres with low latency and high transfer speeds.

Cloud front is integrated with aws- both physical locations that are directly connected to aws global infrstsucture and also the the software that wrks seamlessly with services including , s3 and elb or ec2 and also aws lambda to run custom code close to your viewers.

Cloud front is flexible and abilty to customize.

Pairs nicely witth the existing aws technology

Outperformed all other DDos and cdn providrs.

Benefits:

1. **Global, Growing Content Delivery Network**
2. **Secure Content at the Edge**
3. **Deep Integration with Key AWS Services**
4. **High Performance**
5. **Cost Effective**
6. **Easy to Use**

Cloud frnt servce components: in cloud front we have serveral cpomnts among them two of themare important they are distributions and

1. Distributions : In this we need to what kind of data we are distrbutng that is statc html file or image file or dynamc image file or streaming content.
2. Origin: where is the content cmng from (s3, ec2 server, or onpremises) where we do the html get requests or put requsts are configured in ths
3. Behaviours – it deals all the rules and and it deals with the http requsts and based on these rules we eithr give the cntnt or deny contnt.
4. Restrictions, error , pages,tags: here the data delivering is restrcetd based in the content being served. For expmple if it is the compny data it cannot shared by all the end users so ths data has to restrctd. Here we can also see the geograpc based restrcns, i.e people from certain countries cannot access the data.

We configure the error pages which displays 404 error page.

1. AWS Web application firewall (WAF): scraper protection where some othr tries to steals the data or contnt and ths can be prtctd by scraper prtctn.

SQL injectn protectn:

1. Price classes: You pay accdng to the content being served in diffrnt regns.

Fleet managmnt project –

Fleet in r3- accmdtn , trasnorttn , communication (these are the services refered as fleet)

Accnmndtn – appartmnts town houses, family house

Transpo- cars , mini vans ,

Communication – cell phones,

Fleet management is a platform – combination of serivcs –( Microservices) –

Front end – react

Back – mcroservcs

Microsrevces (built in python )

In lift and sft – microservices(wrtn) - inside – java – springframwrk,springboot

Interntal Local data center – colohub

Th it was mgrted to cloud

Reasons

* no backup
* if the data cntre fails
* database – mongo to dynamo
* in mongodb – it is dfclt to addinstances

**lift and shft –**

part of scrum team of size 10

prdct owner

scrummstr

rest of them are developers(java and python)

two deveops team

exprncd in python and java- some of them

mondo db – driver (mongo s )

in dynamo

tables are called collections

documents -?

install mongo on a mchne

mongo db – has a replica set –

one primary and multiple secondary

in this the driver talk s to prmary and the data is replicated to secondary systms

op log – log file

the data is nt replicated the operatns performd are stored in the log log and ths is performd same for the other systms which can be same as the primary

multiple replica - to balnce the load can be done by sharding

sharding strategy

sharding can be done – range and hash

range :

range mod --- apt no /3 whch gets – 0,1 2

hash – mongo generates md5 #

config server – (maintains all the info ) – we can have a cluster of

ths results in latency to avoid – mongos - cache

* develops roles:
* instance types for the prmary and scndry
* problms
* in dynamo we dnt have the datatypes that are available in mongo and it is dfclt to deal wth it
* for eg – in mongo the lease date – 2017-3-09 this is can be stored by converting it into time stamp in dynamo – which converts it into milliseconds – we have no. of libraries to convert ths into milliscnds

we can use string but during indexing the string does not deal properly wth the string and it wl be easy for indexing in time stamp so we convert the date to tmestamp it is no. type

* run the utility in mongo called – mongo export –whch extrcts to json or csv
* read and wrte capacity in dynamo

we odnt need to maintain the infrstrctre

as we don’t need to scale up and down

no backups