* Route 53
  + DNS (Domain Name System)
    - An address for a website on the internet
  + A highly available and scalable Domain Name System (DNS) web service
  + It is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications by translating website names to numeric IP addresses.
  + Pricing
    - Hosted Zone - 50 cents for every hosted name per month
    - Queries - 40 cents for every million queries
* CloudFront
  + Web service that speeds up distribution of your static and dynamic web content, to your users
  + Delivers content through a worldwide network of data center called edge locations
  + When a request for information is made it is routed to the edge location that provides the lowest latency, so that content is delivered with the best possible performance
  + By delivering web content faster to users it thus provides a better user experience
  + Cloud front edge locations bring the web content closer to your users and make sure that popular content can be served quickly
  + CloudFront regional edge caches really help when the content is not popular enough to stay at a CloudFront edge location and improve delivery performance for that content
  + PRICING
    - <https://aws.samazon.com/cloudfron/pricing/>
    - Price is variable by location
* Applications Load Balancer
  + Achieve fault tolerance for any application by ensuring scalability, performance, and security
  + Automatically distributes incoming application traffic across multiple targets (EC2)
  + Three load balancers
    - Network Load Balancer
      * Protocols: TCP, UDP, TLC
      * Operate at layer 7
    - Classic Load Balancer
    - Application Load Balancer
      * Protocols: HTTP, HTTPS
      * Operate at layer 4
    - <https://aws.amazon.com/elasticloadbalancing/features/>
  + Architecture Components
    - The load balancer is the single point of contact for clients
    - It distributes incoming application traffic across multiple targets, such as ec2 instances, in multiple AZs, which results in increased availability of your application
    - The Listener checks for connection requests from clients, using the protocol/port configured and forwards requests to one or more target groups
    - Define rules for traffic forwarding, including target groups, condition, and priority
    - Target Group
      * Routes requests to one or more registered targets, such as EC2 instances, using protocol/port numbers that you configured
      * A target can be registered with multiple target groups
      * Health checks are run on all targets registered to a TG
* Auto Scaling
  + Helps you ensure that you have the correct number of EC2 instances available to handle the load for your application
  + Ec2 instances are grouped in auto-scaling groups
    - Minimum number of ec2 instances
    - Desired number of ec2 instances
    - Maximum number of ec2 instances
  + Scaling policies will automatically launch or terminate instances as your application demands
  + Launch configurations is an instance configuration template that an ASG uses to launch EC2 instances
    - Includes
      * AMI ID
      * Instance Type
      * Key Pair
      * Security Group
      * Block(Volume)
      * The same as setting up an EC2 instance
  + Auto-Scaling Group
    - Contains a collection of ec2 instances that are treated as a logical grouping for the purposes of automatic scaling and management
    - Maintaining the number of instances in a group and automatic scaling are the core functionalities of the service
    - The size of a group depends on the number of instances you set as the desired capacity
    - It starts by launching the desired number of ec2 instances
    - Create scaling policies to increase or decrease the number of instances in your group dynamically to meet changing conditions
    - The scaling policy adjusts the desired capacity of the group, between the min and max capacity and launches or terminates the instances as needed
* Relational Database Service
  + A web service that makes it easier to set up, operate and scale a relational database in the cloud
  + WHY
    - Easily allocate or increase resources as you need them, on the fly
    - Forget about backups, os patches, recovery
    - Automated or manual backups for DB restore
    - Achieve HA with primary DB and synchronous secondary DB; use read replicas to increase read scaling
    - Control who can access you DB with AWS IAM
  + BUILDING BLOCKS
    - A DB instance which is just a database environment in the cloud
    - Runs a DB engine and AWS Supports:
      * MySQL
      * MariaDB
      * PostgreSQL
      * Oracle
      * Microsoft SQL Server
    - Each engine differ in terms of features; the DB engine controls the DB that it manages
    - DB Instance class
      * Determines the CPU and memory
      * During creation
        + Magnetic
        + General Purpose (SSD)
        + Provisioned IOPS (PIOPS)
      * Each instance has min/max storage requirements; depending on the storage type and DB engine it supports
  + RDS security
    - Implemented through security groups; you can allow access to the DB by specifying Ip address ranges or ec2 instances
    - Three types of security groups
      * DB security group controls access to a DB instance that is not ain a VPC
      * VPC security group controls access to a DB instance inside a VPC
      * EC2 security group controls access of an ec2 instance to the DB
  + INTERFACE
    - Several ways to interface with AWS RDS
      * AWS management console
      * Command Line Interface
      * AWS software development kits
    - Monitoring
      * Amazon CloudWatch service to monitor the performance and health of a DBN instance
      * Performance charts are shown in the RDS console
    - Pricing
      * Clock hours of server time - pay for what you use
      * DB instance type
      * DB purchase type
        + on-demand
        + reserved
      * Number of DBS
      * Backup storage is charged on a per GB/month
      * Number of input and output requests
      * Deployment type
        + Single
        + multi AZs
      * Data transfer - inbound data transfer is free and for outbound data transfer you are charged
* AWS Lambda
  + A compute service that lets you run code without provisioning or managing servers
  + Executes your code only when needed and scales automatically; you pay only for the compute time you consume there is no charge when your code is not running
  + Runs your code on a high-availability compute infrastructure and performs all of the administrative tasks of the compute resources
  + What is happening on the back end:
    - Server and operating system maintenance
    - Capacity provisioning
    - Automatic scaling
    - Code monitoring
    - Logging
  + You can use Lambda to run your code in response to events; as an example, run your code in response to HTTP requests using Amazon API Gateway
  + PRICING
    - Pay only for what you use
    - Pay only for the compute time you consume, there is no charge when your code is not running
    - Charged based on the number of requests for your functions and the time it takes for your code to execute
    - Lambda registers a request each time it starts executing in response to an event notification or invokes a call
* Elastic Beanstalk
  + Quickly deploy and manage applications in the AWS without having to learn about the infrastructure that runs those applications
  + Simply upload your application and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling and application health monitoring
  + Elastic Beanstalk will provision one or more AWS resources, to run your app
  + USE
    - Create an app, upload an app version as a package to the elastic beanstalk and then provide some information about the application
    - Automatically launches an environment and creates and configures the resources needed to run your code
    - The environment is launched, you can then manage your environment and deploy new application versions
  + PRICING
    - There is no additional charge for Elastic Beanstalk usage
    - You only pay for the underlying AWS resources that your application consumes
* CloudFormation
  + Create a template that describes all the resources that you want and aws cloud formation takes care of provisioning and configuring those resources for you
  + You don’t need to individually create and configure aws resources and figure out what’s dependent on what
* Simple Notification Service\
  + A web service that coordinates and manages the sending delivery of messages to subscribing endpoints to clients
  + Two types of clients publishers and subscribers also referred to as producers and consumers
  + Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and comm. Channel
  + Subscribers(ie webservers, emails, sqs queries etc) consume or receive the message or notifications over one of the supported protocols when they are subscribed to the topic
* CloudWatch
  + Monitor AWS resources and the application you run in aws in real-time
  + Create alarms which watch metrics and send notification s or automatically make changes to the resources you are monitoring when a threshold is breached
  + Use this data to also identify and stop underused instances to save money with auto-scaling policies
  + Accessed and used:
    - AWS Cloud Watch console
    - AWS CLI
    - Cloud Watch API
    - AWS SDKs
  + WHERE
    - Auto Scaling EC2
      * Automatically launch or terminate ec2 instances based on user-defined policies
    - CloudTrail
      * Writes log files to the s3 bucket specified when you configured cloud trail
    - Amazon SNS
      * Send messages when an alarm threshold has been reached

REVIEW

* DNS
  + Stands for domain name system and acts as the phonebook of the internet; DNS helps you to resolve names to IP addresses
* Route 53
  + Global, highly available and scalable Domain Name System web service
  + Use to resolve domains and also to register new domains
* Cloud Front
  + A web service that speeds up distribution of your static and dynamic web content for users
  + Delivers content through a worldwide network of data centers called edge locations
  + Regional edge caches really help when the content is not popular enough to sta at an edge location and overall it improves delivery performance for that content
  + The origin is where it gets the information from
  + You use CloudFront when you want to create a distribution and lower latency and increase user experience
* Elastic Load Balancing
  + Achieve fault tolerance for any application by securing scalability performance and security
  + Automatically distributes incoming application traffic across multiple targets
  + Supports the types of load balancers:
    - Network
    - Classic
    - Application
* Auto Scaling
  + Ensures that you have the correct number of instances available to handle the load for your application ( up or down)
  + EC2 instances are grouped in auto-scaling groups:
    - Min number of instances
    - Desired num of instances
    - Max num of instances
  + Scaling policies will automatically launch or terminate instances as your application demands
* RDS
  + Web service that makes it easier to set up, operate and scale relational databases in the cloud
  + Fully managed RDS in the cloud and takes care of all of the work for you
  + A DB instance is just a database environment in the cloud, that runs a DB engine; DBS come in different sizes with different storage options
* Lambda
  + Compute service that lets you run code without provisioning or managing servers
  + Executes code only when needed and scales automatically, pay only for the compute time you consume
  + Run code in response to events such as HTTP requests using the API gateway
* Elastic Beanstalk
  + Quickly deploy and manage applications in the cloud without having to learn about the infrastructure
  + Upload applications and it will handle everything else
  + Provisions different cloud resources to run the application
* CloudFormation
  + Creates a template that describes all the resources that you want and takes care of provisioning and configuring those resources for you
* SNS
  + Web service that coordinates and manages the sending or delivery of messages to subscribing endpoints or clients
  + Publishers communicate async with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel
  + Subscriber consume or receive the message or notification over one of the supported protocols like an email when they are subscribed to the topic
* CloudWatch
  + Monitor your cloud service resources and the applications you run in real-time
  + Create alarms which watch metrics and send notification or make changes based on thresholds set

Fundamentals of Pricing

* Provides agility, and helps reduce your IT costs and reach global coverage in minutes
* Optimize costs continuously, in order to match your needs and environment
* Types of pricings and ROI (return on investment)
  + Pay as you go
  + On-demand
* Key principles
  + Understand the fundamentals of pricing
  + Start early with cost optimization
  + Maximize the power of flexibility
  + Use the right pricing model for the job
* Fundamentals
  + The following create cost impacts
    - Compute (By min or hour)
    - Storage (GB basis)
    - Outbound data transfer (Aggregated as GB)
  + Optimization
    - Never too early to start
    - Start from the beginning of implementation the complexity grows as you move forward and scale your project
    - PUt cost visibility and control mechanisms in place before the environment becomes large and complex.
  + Flexibility
    - Pay exactly for what you need with no minimum commitments or long-term contracts
    - Choose to save money through a reservation model
    - Pay as you go, model, procurement complexity is reduced which enables your business to be fully elastic
    - Don’t pay for services that are not running cost-eff.
  + Price Model
    - Choose the pricing model that bests fits your business needs as well
    - Different models are available for ec2
      * On-demand
        + Pay and use with no upfront payment or long term contract
      * Dedicated instances
        + Hardware is dedicated to you and you don’t split the host with any other customer
      * Spot Instances
        + Purchase spare computing capacity at discounted hourly rates
      * Reservations
        + Pay for computing capacity ahead of time and receive discount up to 75%

Cost Optimization with Reservations

* Achieve significant cost savings by using reserved instances (RIs) and other reservation models for computing and data services
* With RIs you commit in advance for the usage which in return means a lower price
* With reservations, you can choose to pay with no upfront, partial upfront or all upfront
* The larger the upfront payment, the bigger the discount
* Reserve capacity and receive a big discount on your instances, compared to running on an on-demand paying model
* Predict compute costs over the contract term
* To use the capacity you reserved you launch ec2 instances with the configuration as the reserved capacity that you purchased and aws will automatically apply the discounted price.

AWS slogan

* You pay as you go, pay for what you use, pay less as you use more, and pay even less when you reserve capacity
* To estimate monthly bill you can use the simple monthly calculator

Cost Calculators

* Two Options
  + AWS Cost Calc
    - Https: //calculator.aws/
  + AWS Simple Monthly Calc
    - <https://calculator.s3.amazonaws.com/index.html>

Trusted Advisor

* Provides you real-time guidance to help you provision your resources following AWS best practices
* <https://aws.amazon.com/premiumsupport/technology/trusted-adviosr/best-practice-checklist>

Support Plans

* Basic Support (All customers)
  + 24/7 customer service
  + AWS Trusted Advisor
    - 7 core checks and guidance to following best practices
  + AWS Personal Health Dashboard
    - A personalized view of the health of AWS services and alerts when your resources are impacted
* Premium (Developer, Business and Enterprise)
  + Support plans differe in terms of how many “add on services” you get from aws, how much the aws team gets involved in your projects and of course pricing
    - <https://aws.amazon.com/premiumsupport/plans/>

Security

* Introduction to AWS Security
  + Security is AWS top priority and will help you to protect the confidentiality, integrity, and availability of your systems and data
  + AWS architecture has been built following two key principles:
    - Flexibility
    - Security
      * Providing an extremely scalable and flexible cloud platform
  + Redundant and multi-layer controls, continuous validation and testing, with built-in automation, that helps to monitor and keeping customers safe and secure
  + That same level of automation and security is contained and replicated in any of the AWS data centers
  + Get a resilient, fault-tolerant architecture, designed for security, and able to satisfy the requirements of even the most security-sensitive customers.
  + Security and compliance is a shared responsibility between aws and the customer
  + The customer assumes responsibility and management fo the guest operating system (including updates and security patches), as well as the configuration of the AWS, provided security group firewall, while AWS takes care of the cloud
  + This differentiation of responsibility is also known as Security of the cloud versus security in the cloud
* AWS
  + AWS is responsible for protecting the infrastructure that runs all of the services offered in the cloud
  + This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud Services
* Customer Responsibility
  + Customer responsibility will be determined by the AWS Cloud Services that a customer selects
  + This determines the amount of configuration work the customer must perform as part of their security responsibilities ( this means that the customer has to perform more security responsibilities for an EC2 instance than say an elastic beanstalk instance where amazon takes more responsibilities for security)

CUSTOMER

* Customer Data
* Platform, Applications, Identity and access management
* The operating system, Network, and Firewall configuration
* Client-Side Data, Encryption, and Data Integrity and Authentication
* Server-Side Encryption
* Networking Traffic Protection

AWS

* SOFTWARE
  + Compute
  + Storage
  + Database
  + Networking
* Hardware/AWS Global Infrastructure
  + Regions
  + AZs
  + Edge Locations
* <https://aws.amazon.com/compliance/shared-responsibility-model/>

Security Products and Features

* Network Security
  + Built-in firewalls (security groups)
    - Control access to your instances and subnets
  + Encryption in transit using TLS
  + VPNs, for dedicated private connections
  + DDoS mitigation technologies
* Configuration Management
  + Deployment tools for creation and decommissioning of cloud services and resources
  + Inventory tools
    - Dashboards
  + Template definition in order to create custom EC2 instances (specific config that you can replicate)
* Data Encryption (Define encryption at-rest for your data)
  + Data encryption capabilities available for Cloud Storage and DB services
  + Flexible KMS (AWSA or you manage the encryption keys)
  + Hardware-based cryptographic key storage options (sensitive customers)
* Access Control
  + IAM to define individual user accounts with custom permissions
  + MFA (Multi-factored authentication)
  + Integration and Federation with corporate directories
* Monitoring and Logging
  + Deep Visibility
    - CloudTrail (any API call in the environment)
  + Log aggregation
    - CloudWatch
  + Notifications through alerts (emails)
* Security Guidance
  + AWS Enterprise Support
  + AWS Trusted Advisor
  + AWS Professional Services
* AWSA Compliance Program
  + Computing environments are continuously audited, with certifications from accreditation entities across geographies and verticals
  + In a traditional data center, common compliance activities are often manual, periodic activities and include verifying asset configurations and reporting on administrative activities. Moreover, the resulting reports are out of date before they are even published
  + <https://aws.amazon.com/compliance/programs/>

AWS WAF, Shield and Firewall Manager

* AWS WAF
  + A web application firewall that monitors connections forwarded to your web application
  + A WAF protocol is a layer 7 defense (OSI model) and is not designed to defend against all types of attacks
  + As opposed to typical network firewalls, WAFs understand traffic from the application perspective; with WAF you can monitor HTTP and HTTPS traffic which is more than just TCP protocol and 80/443 port numbers
  + WHAT
    - WAF typically protects web applications from attacks such as cross-site forgery, cross-site-scripting(XSS), file inclusion, and SQL injection
    - This method of attack mitigation is usually part of a suite of tools which together create a holistic defense against a range of attack vectors
    - Deploying WAF in front of a web application literally means installing a shield between the web application and the internet users
  + CONFIGURATION
    - Allow all traffic except specific requests
    - Block all traffic except request that you permit
    - Monitor and count requests with properties defined
  + WHY
    - An additional level of security for your web apps
    - Define custom rules to protect web apps
    - Use WAF API for automated administration
* AWS Shield
  + Protects against DDoS attacks
    - A denial-of-service attack is a type of cyber attack in which a hacker aims to make a computer or other server unavailable to its users by interrupting the device’s normal functioning
    - A distributed denial-of-service attack is a DoS attack that comes from many distributed sources
* AWS Firewall Manager
  + Simplifies your AWS WAF administration and maintenance tasks across multiple accounts and resources
  + Only need to set up firewall rules once
  + The service automatically applies your rules across your accounts and resources, even as you add new ones.

AWS Inspector

* Test your instances from the network accessibility perspective and the security state of your applications that run on those instances
* Assesses applications for exposure, vulnerabilities, and deviations from best practices
* Upon completion of an assessment, it will deliver a detailed list of security findings that are organized by level of severity
* HOW
  + You install a small software package, called an agent, in the OS of the instance that you want to assess
  + The agent monitors the behavior of the instance, including network, file system, and process activity

Architecting for the cloud

Intro:

* Migrating applications to AWS, even without significant changes (also known as “lift and shift”), provides organizations the benefits of a secured and cost-efficient infrastructure.
* Architectures need to be changed and get updated, which will lead to immediate benefits lie agility and elasticity, that are possible and available with cloud computing
* Following are the best practices that have emerged, as a result of cloud - traditional computing differences
  + In a traditional data center, resources provisioning is done by guessing and making assumptions on maximum peak load; this results in either idle expensive resources not being utilized or insufficient capacity to handle traffic
  + This is totally different from cloud computing: use the right amount of capacity, dynamically scale up or down when needed, pay-as-you-go and only for what you use
  + AWS services are up and running in minutes, you can use them for as much or as little time as needed, no limits
  + When you deploy your app in the cloud, several best practices should be followed: proximity to you end users, compliance or data residency constraints, costs
  + In order to achieve low latency for your applications, you may want to use Amazon Cloud Front Content Delivery Network
  + High availability and fault tolerance for your apps by using AWS global infrastructure, deploy in multiples DCs
  + With AWS there is virtually unlimited capacity to use
  + AWES services are instantly available to use: compute, storage, databases, analytics, deployment services
  + Using managed services from AWS help you lower operational complexity and also the cost
  + Reducing the risk for you project implementations is easy, as all AWS managed services are designed for scalability and high availability
  + With the cloud, governance capabilities that enable continuous monitoring of configuration changes to you IT resources are always-on and available
  + This is different than traditional infrastructure, where auditing processes are periodic and manual processes.
  + Solution architects can use quite a lot of native AWS security and encryption features and services, which leads to meeting higher levels of compliance and data protection

Tenants:

* Scalability
  + Systems that are expected to grow over time need to be built on top of a scalable architevture
  + Scalable architectures provid ethe ability to grow your enviornment when this is needed (increase n number of users, traffic throughput)
  + Cloud computing allows virtually unlimited growth, but the underlying architecture must be designed to support this
  + Scale either vertically or horizontally
    - Vertically
      * Means increasing the capacity of your current server
      * At some point you discover that you current server can no longer process the amount of data that is constantly increasing you need to scale and grow
    - Horizontally
      * Scaling horizontally means increasing the number of current resources
      * This is not always possible, depending on the underlying architecture, which can or can not distribute traffic to multiple resources
      * EXAMPLES
        + Stateful

There are cases wher you can not change all your components in your architecture to stateless

Eg. MMORPGs

To solve this you can scale using session affinity which binds all connections from a specific user to a single server.

* + - * + Stateless

A stateless application is an application that needs no knowledge of previous interactions and stores no session information. The app will provide the same response to any user with the same input

Stateless applications are great candidates for horizontal scaling; simply add more instances in order to run your app and terminate instances when no longer needed

The easiest way to distribute traffic to the instances is through an elastic load balancer

COMPONENTS

Most applications need to maintain some kind of state information

Some web applications use THHP cookies to store data on the client-side other scenarios require larger files

* + - * + Differences

The key difference between stateful and stateless applications is that stateless applications do not store any data and connections and are independent of one another

* + - * Distributed Processing
        + This is similar to breaking a problem into smaller pieces
        + When a single compute resource can not process that information, because it’s too large for example, then the work will be distributed and split into small fragments and sent to more instances
        + This is the common case for big data scenarios.
* Disposable Resources
  + Temporary resources, easily available, and are one-time use
  + In traditional data center environments you work with fixed resources or server; this translates to you as high upfront costs and a time to production was high as well
  + Within AWS, you launch as many servers as you need to use them as long as you need them and pay accordingly
  + Other Regards to Fixed Servers
    - Configuration drift
      * Configuration changes and software patches can be applied inconsistently; this leads to different configurations on your resources in the DC
    - Immutable infrastructures
      * This can solve the previous issue; instead o patching and modifying initial configuration on your servers, just change the old server with a new one that has new software packages applied
    - Infrastructure Instantiating
      * Manually setting up your infrastructure is time-consuming and is also error-prone
      * Ideally, any new environment setup or scaling up existing infrastructure should be done automatically
        + AWS provides bootstrapping or golden images as solutions
  + Bootstrapping
    - Provide a startup string of startup commands to get everything configured when an instance is launched so that scaling images is trivial
  + Golden Image
    - If you have an instance that is up and running with all security and OS patches applied you can create a snapshot of this instance
    - The snapshot (golden image) may then be used in order to create an AMI
    - The AMI could be used for example in an Auto Scaling group so that resources sustaining you app can scale up or down as needed
* Automation
  + Less manual work, less possible errors, improve the system’s stability and efficiency
  + AWS Elastic Beanstalk
    - Upload application code and provisioning, load balancing, auto-scaling, and monitoring are done automatically.
  + Amazon Instance Auto Recovery
    - Monitor an instance and if it fails it creates an identical instance
  + Auto Scaling
    - Scale instances capacity up or down depending on the conditions you define in the auto-scaling policy
  + ClouWatch Alarms:
    - Define alarms that can trigger other actions as well
  + Lambda Scheduled Events:
    - Execute a task at a specific time of day
  + Ops Works Lifecycle Events
    - Supports continuous configuration through events; i.e. update you instances configuration as a result of an event (trigger)
* Loose Coupling
  + Breaking your application into smaller pieces or components in such a way that they are little to not dependent on each other leading to a loose coupling system
  + IMPLEMENTATION
    - Communication between the components should be implemented through open-source mechanisms
    - Using open-source communication interfaces leads to the possibility of developers to modify and adapt configurations on the fly during or after project implementations
    - Implementation means that you will have a lot of services that need either communicate with each other or with other services in your environment
    - There needs to be a way to address or call any service in a unique way so that no interdependencies are created
    - An example is load balancers you can call a load balance by using the endpoint name
  + ASYNC INTEGRATION
    - Refers to integrations between different services in your infrastructure
    - ASYNC
      * Tow services can work independently of each other, but together as a system, this means that the system is asynchronous
  + Graceful Failure
    - When a failure occurs, communication of the failure should be performed into the system and all components should be aware
    - Rerouting of traffic to healthy services should take place and route 53 has the capability to do this
* Services
  + The main idea is that you should let AWS managed services and migrate to serverless architectures as much as possible
  + Managed services, means services that are managed making your life easier. This includes databases, machine learning, analytics, queuing, email, notifications, and etc.
  + SERVERLESS
    - Lambda is the compute service that will run your code on your behalf without needing to set up a server. It is also cheaper than traditional cloud computing
    - Lamda you are charged for every 100ms your code executes and the number of times your code is triggered
* <https://dl.awsstatic.com/whitepapers/AWS_Cloud_Best_PRactices.pdf>