AWS Fundamentals

AWS Regions:

* AWS has regions all around the world
* Names can be: us-east-1 etc
* A region is a cluster of data centers
* Most AWS services are not region scoped

AWS Availability Zones

* Each region has many availability zones with a min of 2 to max of 6
* Each AZ is one or more discrete data centers with redundant power, networking , and connectivity
* They’re separate from each other, so that they’re isolated from disasters
* They’re connected with high bandwidth ultra-low latency networking

IAM Introduction

* IAM (Identity and Access Management)
* Your whole AWS security is there:
  + Users - Physical person
  + Groups - function, teams contains users
  + Roles - Internal usage with AWS resources
  + Policies - All of the above use policies
* Root account should never be used ( and shared)
* Users must be created with proper permissions
* IAM is at the center of AWS
* Policies are written in JSON
* IAM has a global view
* Permissions are governed by policies (JSON)
* MFA (Multi Factor Authentication) can be setup
* IAM has predefined “managed policies”
* We’ll see IAM policies in details in the future
* It’s best to give users the minimal amount of permissions they need to perform their job ( least privilege principles)

IAM Federation

* Big enterprises usually integrate their own repository of users with IAM
* This way, one can login into AWS using their company credentials
* Identity Federation uses the SAML standard (Active Directory)

IAM 101 Brain Dump

* One IAM User per physical person
* One IAM Role per application
* IAM credentials should never be shared
* Never, ever, ever, ever write IAM credentials in code. Ever
* And even less never ever ever commit your IAM credentials
* Never use the root account except for initial setup
* Never use root iam credentials

AWS EC2

* EC2 is one of most popular of AWS offering
* It mainly consists in the capability of:
  + Renting virtual machines (EC2)
  + Storing data on virtual drives (EBS)
  + Distributing load across machines (ELB)
  + Scaling the services using an auto-scaling group(ASG)
* Knowing EC2 is fundamental to understand how the cloud works

Introduction to Security Groups

* Security Groups are the fundamental of network security in AWS
* They control how traffic is allowed into or out of our EC2 Machines
* It is the most fundamental skill to learn to troubleshoot networking issues

Security Groups Deeper Dive

* Security groups are acting as a “firewall” on EC2 instances
* They regulate:
  + Access to Ports
  + Authorised IP ranges – IPv4 & IPv6
  + Control of inbound network ( from other to the instance)
  + Control of outbound network (from the instance to other)
  + Classless Inter-Domain Routing **CIDR***\* Taken from wikipedia \**
    - A method for allocating IP addresses and for IP routing.
    - Its goal was to slow the growth of routing tables on routers across the Internet, and to help slow the rapid exhaustion of IPv4 addresses
    - Based on variable-length subnet masking (VLSM) which allow sthe specifications of arbitrary-length prefixes. CIDR introduced a new method of representation for IP addresses, now commonly known as CIDR notation in which an address or routing prefix is written with a suffix indicating the number of bits of each prefix.
      * IE: 192.0.2.0/24 for IPv4
      * IE: 2001:db8::/32 for IPv6
    - CIDR introduced an administrative process of allocating address blocks to organizations based on their actual and short term projected needs.
    - The aggregation of multiple contiguous prefixes resulted in supernets in the larger Internet, which whenever possible are advertised as aggregates, thus reducing the number of entries in the global routing table.
* They do not regulate
  + Domain Name

Security Groups Good to Know

* Can be attached to multiple instances
* Locked down to a region / VPC combination
* Does live “outside” the EC2 – if traffic is blocked the EC2 instance won’t see it
* It’s good to maintain one separate security group for SSH access
* If your application is not accessible ( time out), then it’s a security group issue
* If your application gives a “connection refused” error, then it’s an application error or it’s not launched
* All inbound traffic is blocked by default
* All outbound traffic is authorised by default

Referencing other security groups

* If security groups are shared by ec2 instances they can communicate with each other regardless of IP address

Private vs Public IP (IPv4)

* Networking has two sorts of IPs. IPv4 and IPv6
  + IPv4: 1.160.10.240
  + IPv6: 3ffe: 1900:4545:3:200:f8ff:fe21:67cf
* IPv4 is still the most common format used online.
* IPv6 is newer and solves problems for the Internet of Things ( IoT).
* IPv4 allows for 3.7 billion different adddresses in the public space
* IPv4: [0-255].[0-255].[0-255].[0-255]

Private vs PUblic IP Fundamental Differences

* Public IP:
  + Public IP means that machines can be identified on the internet
  + Must be unique across the whole web (meaning there are no two same public IPs)
  + Can be geo-located easily
* Private IP:
  + Private IP means the machine can only be identified on private network only
  + The Ip must be unique across the private network
  + But two different Private networks ( two companies) can have the sam IPs
  + Machines connect to WWW using a NAT + internet gateway (a proxy)
  + Only a specific range of IPs are allowed for Private IPs

Elastic IPs

* When you stop and then start an EC2 instance, it can change its public IP.
* If you need to have a fixed public IP for your instance, you need an Elastic IP
* An Elastic IP is a public IPv4 IP you own as long as you don’t delete it
* You can attach it to one instance at a time
* With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account
* You can only have 5 Elastic IP in you account \* you can ask AWS to increase that).
* Overall, try to avoid using Elastic IP:
  + They often reflect porr architectural decisions
  + Instead, use a random public IP and register a DNS name to it
  + Or, use a Load Balancer and don’t use a public IP

EC2 by Default

* Comes with
  + A private IP for the internal AWS Network
  + A public IP, for the WWW.
* SSH into EC2
  + We can’t use a private IP , because we are not in the same network
    - Unless you vpn into the network
  + We can only use the public IP.
* If your machine is stopped and then started the public IP can change

EC2 User Data

* It is possible to bootstrap our instances using an EC2 User data script
* Bootstrapping means launching commands when a machine starts
* That scripts is only run once at the instance first start
* EC2 user data is used to automate boot tasks such as:
  + Installing updates
  + Installing software
  + Downloading common files from the internet
  + Anything you can think of
* The EC2 User Data Script runs with the root user

EC2 Instance Launch Types

* On Demand Instances: short workload, predictable pricing
* Reserved: ( min 1 year)
  + Reserved Instances: long workloads
  + Convertible Reserved Instances: long workloads with flexible isntances
  + Scheduled Reserved Instances: example – every Thursday between 3 - 6pm
* Spot Instances: short workloads, for cheap , can lose instances ( less reliable)
* Dedicated Instances: no other customers will share your hardware
* Dedicated Hosts: book an entire physical server, control instance placement

EC2 On Demand

* Pay for what you use ( biling per second after the first minute )
* Has the highest cost but no upfront payment
* No long term commitment
* Recommended for short-term and un-interrupted workloads, where you can’t predict how the application will behave.

EC2 Reserved Instances

* Up to 75% discount compared to on demand
* Pay upfront for what you use with loong term commitment
* Reservation periods can be 1 or 4 years
* Reserve a specific instance type
* Recommended for steady state usage applications ( think database )
* Convertible Reserved Instance
  + Can change the EC2 instance type
  + Up to 54% discount
* Scheduled Reserved Instances
  + Launch within time window you reserve
  + When you require a fraction of day / week / month
* EC2 Spot Instances
  + Can get a discount of up to 90% compared to On-demand
  + Instances that you can “lose” at any point of time if you max price is less than the current spot price
  + The most cost efficient instance in AWS
  + Useful for workloads that are resilient to failure
    - Batch jobs
    - Data analysis
    - Image processing
    - …
  + Not great for critical jobs or databases
* Great combo: Reserved instances for baseline + On-Demand & spot for peaks

EC2 Dedicated Hosts

* Physical dedicated EC2 server for your use
* Full control of Ec2 Instance placement
* Visibility into the underlying sockets / physical cores of the hardware
* Allocated for your account for a 3 year period reservation
* More expensive
* Useful for software that have complicated licensing model ( BYOL – Bring you own license)
* Or for companies that have strong regulatory or compliance needs

EC2 Dedicated Instances

* Instances running on hardware that’s dedicated to you
* May share hardware with other instance in same account
* No control over instance placement(can move hardware after Stop/Start)

EC2 SPot Instance Requests

* Can get a discount of up to 90% compared to On-demand
* Define max spot price and get the instance while current spot price < max
  + The hourly spot price varies based on offer and capacity
  + If the current spot price > your max price you can choose to stop or terminate your instance with a 2 minutes grace period
* Other strategy: Spot Block
  + “Block” spot instance during a specified time frame ( 1 to 6 hours) without interruptions
  + In rare situations, the instance may be reclaimed
* Used for batch jobs, data analysis, or workloads that are resilient to failures
* Not great for critical jobs or databases

How to terminate Spot Instances

* Two types of requests ( one-time | persistent )
* one-time
  + Will launch the instances and when interrupted will close the instance permanently
* Persistent
  + They will continually be launched and interrupted based on the spot request protocol. To cancel them The ec2 has to be in the open, active, or disabled states

Spot Fleets ( the ultimate way to save money)

* Spot Fleets = set of Spot Instances + (optional) On-Demand Instances
* The spot Fleet will try to meet the target capacity with price constraints
  + Define possible launch pools: instance type, OS, AZ
  + Can have multiple launch pools, so that the fleet can choose
  + Spot Fleets stops launching instances when reaching capacity or max cost
* Strategies to allocate Spot Instance:
  + lowestPrice: from the pool with the lowest price ( cost optimization, short workloads)
  + Diversified: distributed across all pools ( great for availability, long workloads)
  + Capacity Optimized: Pool with the optimal capacity for the number of instances
* Spot Fleets allows us to automatically request Spot Instances with the lowest price

EC2 Instance Types – Main ones

* R: applications that needs a lot of RAM – in-memory caches
* C: applications that needs good CPU – compute / databases
* M: applications that are balanced ( think “ medium” ) – general / web app
* I: applications that need good I/O (instance storage) – databases
* G: applications that need a GPU – video rendering / machine learning
* T2 / T3: burstable instances ( up to a capacity)
* T2 / T3 - unlimited: unlimited burst
* Real-world tip: use <https://www.ec2instances.info>

Burstable Instances (T2/T3)

* AWS has the concept of burstable instance ( T2/T3 machines)
* Burst means that overall, the instance has OK CPU performance
* When the machine needs to process something unexpected ( aspike in tload for example), it can burst, and CPU can be verry good
* If the machine bursts, it utilizes “ burst credits”
* If all the credits are gone, the CPU\* becomes BAD
* If the machine stops bursting, credits are accumulated over time

Burstable Instances (T2/T3)

* Burstable instances can be amazing to handle unexpected traffic and getting hte insurance that it will be handled correctly
* If your instance consistently run low on credit, you need to move to a different kind of non-burstable instance

T2/T3 Unlimited

* Nov 2017: It is possible to have an “unlimited burst credit balance”
* You pay extra money if you go over your credit balance, but you don’t lose in performance
* Overall , it is a new offering , so be careful, costs could go high if you’re not monitoring the health of you instances

Why would you use a custom AMI

* Using a custom built AMI can provide the following advantages:
  + Pre-installed packages needed
  + Faster boot time ( no need for ec2 user data at boot time)
  + Machine comes configured with monitoring/ enterprisesoftawware
  + Security concerns– control over the machines in the nework
  + Contorl of maintenance and updates of AMIs over time
  + Active Directory Integrations out of the bod
  + Installing your app ahead of time ( for faster deploys when auto-scaling)
  + USing someone else’s AMI that is optimised for running for a specific purpose
* AMI are built for a specific AWS region (!)

Using PUblic AMIs

* YOu can leverage AMIs from other people
* You can also pay for other people’s AMI by the hour
  + These people have optimised the software
  + The machine is eady to run and configure
  + YOu basically rent “expertise” from teh AMI creator
* AMI can be found and published on the amazon marketplace
* WARNING
  + Do not use an AMI you don’t trust!
  + Some AMIs might come with malware or may not be secure for your enterprise

AMI Storage

* Your AMI take space and they live in Amazon S3
* Amazon s3 is a durable, cheap and resilient storage where most of your backups will live ( but you won’t see them in the S3 console)
* By default, your AMIs are private, and locked for your account / region
* You can also make your AMIs public and share them with other AWS acounts or sell them on the AMI Marketplace

AMI Pricing

* AMIs live in Amazon S3 so you get charged fro the actual space in takes in Amazon s3
* Overall it is quite inexpensive to store private AMIs.
* Make sure to remove the AMIs you don’t use

Cross Account AMI Copy (FAQ + Exam Tip)

* You can share an AMI with another AWS account.
* Sharing an AMI does not affect the ownership of the AMI
* If you copy an AMI that has been shared with your account, you are the owner of the source AMI must be granted you read permissions for the storage that backs the AMI, either the associated EBS snapshot ( for an AMazon EBS backed AMI) or an associated S3 bucket ( for an instance store-backed AMI).
* Limits:
* YOu can’t copy an encrypted AMI that was shared with you from another account. Instead if the underlying snapshot and encryption key were shared with you you can copy the snapshot while re-encrypting it with a key of your own. You own the copied snapshot, and can register it as a new AMI.
* You can’t copy an AMI with an associated billingProduct code that was shared with you from another account. This includes Windows AMIs and AMIs from the AWS Marketplace. To copy a shared AMI with a billing Product code, launch an EC2 instance in your account using the shared AMI and then create an AMI from the instance.

Placement Groups

* Sometimes you want control over the Ec2 instance placement strategy
* That strategy can be defined using placement groups
* When you create a placement group, you specify one of the following strategies for the group:
  + Cluster – clusters instances into a low-latency group ina single Availability Zone
  + Spread – spreads instances across underlying hardware –critical applications
  + Partition – Spreads instances across many different partitions ( which rely on different sets of racks) within an AZ. Scales to 100s of EC2 instances per group

Cluster

* Pros: Great network (10Gbps bandwidth between instances)
* Cons: If the rack fails, all instances fails at the same time
* Use Case:
  + Big Data job that needs to complete fast
  + Application that needs extremely low latency and high network throughput

Spread

* Pros:
  + Can span across Availability Zones
  + Reduced risk is simultaneous failure
  + EC2 Instances are on different physical hardware
* Cons
  + Limited to 7 instances per AZ per placement group\
* Use Case
  + Application that needs to maximize high availability
  + Critical Applications where each instance each instance must be isolated from failure from each other

Partition:

* Up to 7 partitions per AZ
* Up to 100s of EC2 instances
* The instances in a partition do not share racks with the instances in the other partitions
* A partition failure can affect many EC2 but won’t affect other partitions
* EC2 instances get access to the partition information as metadata
* Use Cases: HDFS, HBase, Cassandra, Kafka

Elastic Network Interfaces

* Logical component in a VCP that represents a virtual network card
* The ENI can have the following attributes:
  + Primary private IPv4, one or more secondary IPv4
  + One Elastic IP (IPv4) per private IPv4
  + One Public IPv4
  + One or more security groups
  + A MAC address
* You can create ENI independently and tatach them on the fly ( move them) on EC2 instances for failover
* Bound to a specific availability zone
* Additional Reading: <https://aws.amazon.com/blogs/aws/new-elastic-network-interfaces-in-the-virtual-private-cloud/>

EC2 Hibernate

* We know we can stop, terminate instances
  + Stop: the data on disk (EBS) is kept intact in the next start
  + Terminate: any EBS volumes (root) also setup to be destroyed is lost
* On start, the following happens:
  + First start: the OS boots & the EC2 User Data script is run
  + Following starts: the OS boots up
  + Then your application starts, caches get warmed up, and that can take a fair amount of time
* Introducing EC2 Hibernate:
  + The in-memory (RAM) state is preserved
  + The instance boot is much faster! ( the OS is not stopped / restarted)
  + Under the hood: the RAN state is written to a file in the root EBS volume
  + The root EBS volume must be encrypted
* Use Cases:
  + Long-running processing
  + Saving the RAM state
  + Services that take time to initialize
* Good to know
  + Supported instance families
    - C3, C4, C5, M3, M4, M5, R3, R4, and R5
    - Instance RAM size - must be less than 150 GB
    - Instance size - not supported for bare metal instances
    - AMI - Amazon Linux 3, Linux AMI, Ubuntu.. & Windows
    - Root Volume: must be EBS, encrypted, not instance store, and large
    - Available for On-Demand and Reserved Instances
  + An instances cannot be hibernated for more than 60 days

EC2 for Solutions Architects

* EC2 instances are billed by the second, t2.micro is free tier
* On Linux / Mac we use SSH, on Windows we use Putty
* SSH is on port 22, lock down the security group to your IP
* Timeout issues => Security group to your IP
* Permission Issues on the SSH => run “chmod 0400”
* Security Groups can reference other Security Groups instead of IP ranges ( very popular exam question)
* Know the difference between the Private, Public, and Elastic IP
* You can customize an EC2 instance at boot time using EC2 User Data
* Know the 4 EC2 launch modes:
  + On demand
  + Reserved
  + Spot instances
  + Dedicated Hosts
* Know the basic instance types: RCMIGT2/T3
* You can create AMIs to pre-install software on your EC2 => faster boot
* AMI can be copied across regions and accounts
* EC2 instances can be started in placement groups:
  + Cluster
  + Spread
  + Partition

Scalability & High Availability

* Scalability means that an application / system can handle greater loads by adapting
* There are two kindle scalability:
  + Vertical Scalability
  + Horizontal Scalability ( = elasticity)
* Scalability is linked but different to High Availability

Vertical Scalability

* Vertically scalability means increasing the size of the instance
* Scaling that application vertically means increasing the EC2 size
* Vertical scalability is very common for non distributed systems, such as a database
* RDS, Elasticache are services that can scale vertically
* There’s usually a limit to how much you van vertically scale because you hit a hardware limit

Horizontal Scalability

* Horizontal Scalability means increasing the number of instances / systems for your application
* Horizontal scaling implies distributed systems
* This is very common for modern applications
* It’s easy to horizontally scale thanks to the cloud offerings such as Amazon EC2

High Availability

* High Availability usually goes hand in hand with horizontal scaling
* High availability means running your application / system in at least 2 data centers 9== Availability Zones)
* The goal of high availability can be passive ( for RDS Multi AZ for example)
* The high availability can be active ( for horizontal scaling)

High Availability & Scalability for EC2

* Vertical Scaling: increase Instance size ( = scale up / down)
* Horizontal Scaling: Increasing the number of instances (= scale out/in )
  + Auto Scaling Group
  + Load Balancer
* High Availability: Run instances for the same application across multi AZ
  + Auto Scaling Group Multi AZ
  + Load Balancer Multi AZ

What is load balancing

* Load balancers are servers that forward internet traffic to multiple servers (EC2 instances) downstream

Why use a load balancer

* Spread load across multiple downstream instances
* Expose a single point of access (DNS) to your application
* Seamlessly handle failures of downstream instances
* Do regular health checks to your instances
* Provide SSL termination (HTTPS) for your websites
* Enforce stickiness with cookies
* High availability across zones
* Separate public traffic from private traffic

Why use an EC2 Load Balancer?

* An ELB (EC2 Load Balancer) is a managed load balancer
  + AWS guarantees that it will be working
  + AWS takes care of upgrades, maintenance, high availability
  + AWS provides only a few configuration knobs
* It costs less to setup your own load balancer but it will be a lot more effort on your end.
* It is integrated with many AWS offerings / services

Health Checks

* Health Checks are crucial for Load Balancers
* They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
* The health check is done on a port and a route (/health is common)
* If the response is not 200 (OK), then the instance is unhealthy

Types of load balancer on AWS

* AWS has 3 kinds of managed load balancers