AWS Whitepaper Notes

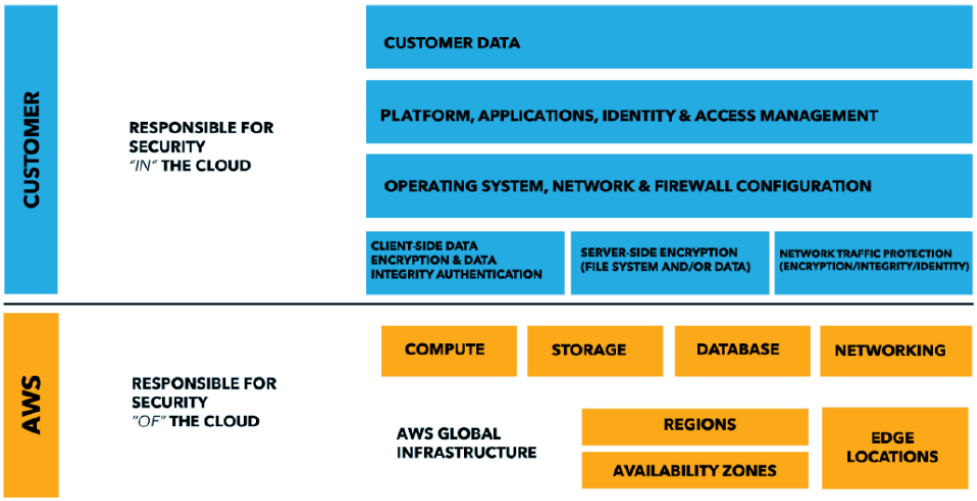
Contents

AWS Security – Introductions*

To be added later.

AWS Security – Overview Process*

AWS Share Responsibility Model: Anything that put in the cloud customer is responsible for it, security of the cloud is the responsibility of the AWS. The amount of security varies between service and the sensitivity of the data. For all service basic security features are common like – SSL/TSL for data transmission, user activity monitoring etc. AWS takes care of the AWS cloud infrastructure and the security for its managed services like lambda functions, Amazon RDS, etc. For Managed services AWS takes care of the operating system patching, database patching, firewall configuration, and disaster recovery.



For customer, if they are availing IaaS service like EC2 instance then customers are responsible for

- Management of the guest operating systems (including update and security patching)
- Installation of the application utility and services
- configuration of AWS provided security firewall called as security group.
- customer is involved in the same task as they are managing their own servers irrespective of where it's been installed.

For customer, if they are availing SaaS (AWS Managed services)
- AWS handles most of the responsibilities on behalf of the customers it takes care of the launching of the AWS manage services instances, patching of the guest operating system or databases, replicating of the database.
- Customer is responsible for managing user access and permissions for proper segregation of duties.

AWS Security Compliance: For AWS compliance responsibilities are also shared between customer and the AWS. AWS platform is aligned to the following security compliance standards

- SOC 1/SSAE 16/ISAE 3402 (formerly SAS 70)
- SOC 2
- SOC 3
- FISMA, DIACAP, and FedRAMP
- DOD CSM Levels 1-5
- PCI DSS Level 1
- ISO 9001 / ISO 27001
- ITAR
- FIPS 140-2
- MTCS Level 3

Along with the above list, AWS is also compliant of the following industry-specific compliance standards

- Criminal Justice Information Services (CJIS)
- Cloud Security Alliance (CSA)
- Family Educational Rights and Privacy Act (FERPA)
- Health Insurance Portability and Accountability Act (HIPAA)
- Motion Picture Association of America (MPAA)

Along with the above mentioned global and industry specific standards, AWS also provides wide range of whitepaper, reports, certifications, and other third-party attestation which can be download from the AWS site.

Physical Security and Environmental Security: AWS provide adequate security to protect Physical AWS datacenters – ONLY in need basis with minimum of two-factor authentications. AWS following protection for its own datacenters

1. **Fire Detection and suspensions**
2. **Power** – AWS always mating a two redundant power supplies**.**
3. **Climate and Temperature control.**
4. **Management**
5. **Storage device decommissioning:** Once a hardware (storage device) reach its end of life, AWS usages NIST (800-880) guidelines "Guidelines for Medial Sanitization" for decommissioning of the storage devices.

AWS Best Practices – Security*

AWS Security – Lambda service overview*

AWS Best Practices – Microservices

Microservice architecture is not a completely a new approach of software development instead it's a combination of various successful and proven concepts such as:

- Agile Application Development
- Service Oriented Architecture
- API first approach.
- Continuous Integration and Continuous Delivery.
- 12 factor app, design pattern.

Microservice can be build using AWS Serverless technology, which along with various other benefits also leverage the benefits of the serverless framework like

- No infrastructure to provision or managed
- Automatically scale by unit of consumption.
- "*Pay for Value*" billing model.
- "Build in availability and fault tolerance"

Unlike traditional approaches, where monolithic application is built as layers of service. In case of microservices architecture, the functionalities are spitted into cohesive verticals, align as per their functional domain.

Microservice UI are mostly developed as single page JavaScript application hosted from Amazon S3 services, while static content is made available through Content Delivery Network (CDN) to reduce latency. To reduce chattiness and improve latency additionally caching mechanism can be built to cache frequently accessible items.

In AWS platform Microservice implementation are most done through AWS Lambda or through docker container with AWS Farget. AWS Lambda function helps quickly develop API to cater business requirements without worry much about the server provisioning. Another common approach is to host docker containers over AWS Farget. To reduce complexity of cluster management of the AWS Farget , one can leverage AWS ECS (Elastic Container Service) or AWS EKS (Elastic Kubernetes Service) where with help of a single API invocation one can launch or stop a Docker enable application and at the same time leverage some of the existing AWS services like ELB (Elastic Load Balancer) , EBS (Elastic Block Store), or IAM (Identity and access management).

AWS Farget can automatically support scaling of resources by launching thousands of containers to cater the incoming load. It also supports container placement strategy and termination task.

AWS EKS – is a managed service which run latest version of Kubernetes Service and all available plugins & tooling available from the Kubernetes communities, at the same time supports IAM feature to securing the application. Application running on any standard Kubernetes environment is fully compatible with the application running on AWS EKS.

Amazon Elastic Container Repository helps in storing docker image use in AWS ECS or in AWS EKS without any need for provisioning and hosting servers for container repository.

AWS Private Link helps in establishing connection withing the VCP and Private Link Supported AWS services (S3/Dynamo DB), AWS services hosted in separate AWS account and supported Marketplace partnered services without any need for Direct Connect, Internet Gateway, NAT devices, Public IPs. The traffic pass through AWS Private Link remains within AWS network & does not leave the AWS network. Private link is a great way to isolate microservices where each microservices can be hosted over a separate VPCs, scale is as per the need of the customer and make it available through AWS Private Link directly or through AWS Marketplace.

There are many options for data storage for microservice – AWS RDS for structural data storage, for high throughput application AWS DynamoDB can be used which is a NOSQL database. There are multiple options available for caching where frequently accessible data can be store to improve performance. DAX (DynamoDB Accelerator, provide caching in between the application and the DynamoDB.

Dynamo DB provide serverless offering, where read-write capacity need not to be guess in advance. Alternatively, one can also use the on-demand Dynamo DB.

API Implementation

Architecting, deploying, monitoring, continuously improving and maintaining an API is a tedious task. AWS API Gateway helps in hosting APIs without any need for provisioning backend servers for hosting APIs. AWS Lambda function can be integrated with the API gateway to develop microservice APIs, without any need for provisioning /hosting / scaling backend server resources.

Deploying microservice using docker image over AWS Farget without worrying about the underline infrastructure. In additionally to this for data storage one can use serverless AWS DynamoDB and AWS Aroura DB.

Deploying AWS Lambda function

For deploying AWS Lambda function, one can leverage AWS SAM (Serverless Application Module) which provides an easy way to define AWS Lambda function and other serverless resources. AWS SAM also provide SAM local which can be used to develop, analyzed and test Lambda function locally before deploying them into AWS Cloud infrastructure.

Distributed System

There are multiple challenges for a distributed system, like service discovery, data consistency, asynchronous communication, distributed monitoring and logging.

Service Discovery:

One of the most challenging tasks of a distributed system is service discovery. Along with the service discovery one also needs to needs to decide how to store service metadata, how to identify healthy services etc. AWS provided following option for service discoveries:

DNS Base Service Discovery:

Amazon ECS can be integrated with the Route53 base service discovery service (Route53 Auto Naming API) which helps discovery of services. Service name are automatically mapped to the Route53 DNS records – client service can use the service name to resolve server endpoint using DNS queries. One can also specify health check condition in the service task definition so that ECS returns ONLY the healthy service endpoints in response to the DNS query.

Additionally, one can also leverage AWS CloudMap capabilities that extend Route53 Auto Naming feature and provide API base service discovery mechanism with faster change propagation.

For services hosted in AWS EKS one can leverage unified service discovery for services hosted in AWS EKS.

Alternatively, Leveraging Third Party Software for service discovery is also possible in AWS – third party software like HashiCorp Consul, etcd or Netflix Eureka can be hosted within AWS infrastructure. AWS Quick Start supports launching of flexible, scalable HashiCorp Consul automatically.

App Mesh: To ease out the communicating between larger number of services, one can use AWS APP Mesh which creates an additional layer for handling inter-service communication. AWS Service Mesh handles the service discovery transparently, application developer need not to add additional application codding for service communication. AWS Service Mesh standardized the communication, adding end-to-end visibility to the service while increasing the availability of the service. APP Mesh can be integrated with existing or new microservices running on AWS Fargate, Amazon ECS, Amazon EKS or on self-managed Kubernetes on AWS.

Distributed data storage

Unlike in the case of monolithic application where application data are generally stored in a relational database with a common data model for all application layers. In case of the microservices there is a need for distributed data storage – each microservices should have their own data persistence layer. However, distributed data management pose few challenges.

In case of distributed data management system, among Consistency, Availability and Partition ONLY two can be achieved at a given time (CAP Theorem). Also, in case of a microservice the business transactions span across microservice making it difficult to implement AICD transaction, instead one needs to use multiple local transaction within microservices to redo a already processed transactions , in case the business transaction fails multiple local transaction needs to be perform to undo the successful transactions (SAGA pattern). *AWS Step Function* helps in SAGA patterns with ease.

For building centralized reference data, AWS lambda functions with schedule Cloud Watch Events can be used to clean up and manage duplication.

In context of microservice architecture, event sourcing enables decoupling of the services (different services of a same application) using publish/subscribe pattern where same event can be feed to different services with different data mode. This can be used in conjunction with the CQRS (Command Query Responsibility Segregation) where read workload is decoupled from the write workload in order to optimized performance, scalability and security. Amazon Kinesis Data Stream can be use to build centralized event store for enabling event sourcing within microservice running on different AWS platforms.

Asynchronous communication & Lightweight Messaging

In case of traditional monolithic application, the communication within different components is straight-forward, whereas in case of microservices communication within different components of a same application needs to pass through the network.

REST-Based communication

Most cases, a REST Based HTTP/S communication is preferred to communicate between different components of a microservice application. The REST architectural style relies on stateless communication, uniform interfaces, and standard methods. API Gateway can used as a front door for the backend APIs. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management.

Asynchronous Messaging and Event Processing:

In this type of approach, the communication between two microservice is achieve through a queue. Benefit of this type of communication is, there is NO NEED for service discovery and the services are loosely coupled preventing fan out failure to the downstream systems while making retries easy. AWS offers multiple services like Amazon SNS, Amazon SQS and Amazon MQ to implement asynchronous messaging.

Orchestration and State Management:

AWS offers AWS Step Functions service, which helps in developing an orchestration layer for distributed system. Step Functions provides a state machine that hides the complexities of service orchestration, such as error handling and serialization/parallelization. This also helps in scaling and make change quickly without any need to change the logic of underline microservices.

Step Functions is part of the AWS serverless platform and supports orchestration of Lambda functions as well as applications based on compute resources, such as Amazon EC2 and Amazon ECS, and additional services like Amazon SageMaker and AWS Glue.

Distributed monitoring:

In case of distributed system, monitoring each individual component is critical at the same time difficult to achieve because of the distributed nature of the system.

Monitoring

Amazon CloudWatch to collect and track metrics, centralize and monitor log files, set alarms, and automatically react to changes in your AWS environment. CloudWatch can monitor AWS resources such as EC2 instances, DynamoDB

tables, and RDS DB instances, as well as custom metrics generated by the applications and the services or any log files that applications generate.

Another popular option–especially for Amazon EKS–is to use *Prometheus*. Prometheus is an open-source monitoring and alerting toolkit that is often used in combination with *Grafana* to visualize the collected metrics. Many Kubernetes components store metrics at /metrics and Prometheus can scrape these metrics at a regular interval.

Centralized Logging

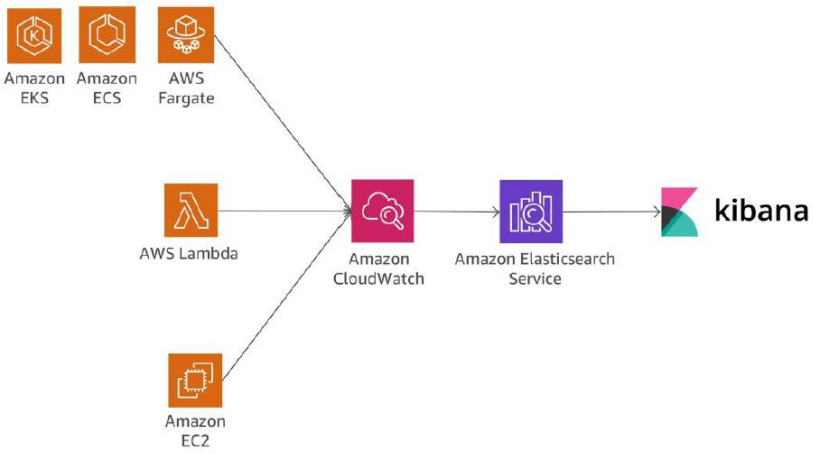
For serverless application its important to log in a centralized location. AWS CloudWatch provides an easy way to log centrally, by default lambda logs are store in AWS CloudWatch. AWS ECS provides support for awslogs log driver to route container log to AWS CloudWatch. For Amazon EKS, it is necessary to run FluentD which forwards logs from the individual instances in the cluster to a centralized logging CloudWatch Logs where they are combined for higher-level reporting using Elasticsearch and Kibana.

Distributed Trace

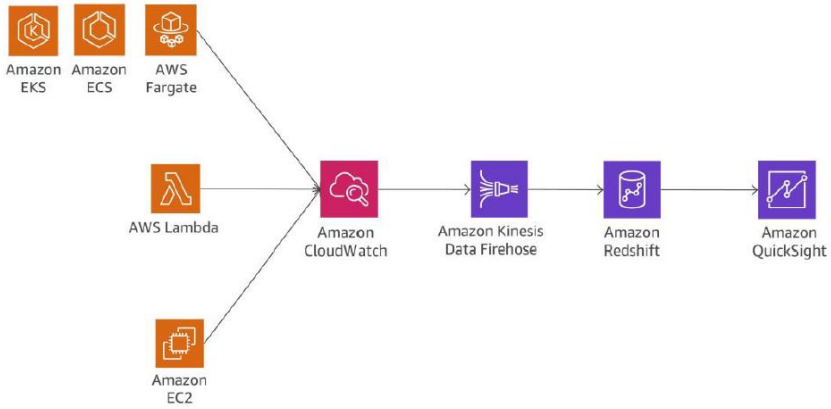
In case of microservice application as there are multiple services so its equally important to track the end-to-end logs for a business flow across all microservices. AWS X-Ray provides a single correlation ID which unique identifiers and attached to all requests and messages related to a specific event chain. The trace ID is added to HTTP requests in specific tracing headers named `X-Amzn-Trace-Id` when the request hits the first X-Ray-integrated service. With the help of correlation ID & trace ID, it's possible to track end-to-end business flow.

Log Analysis

Searching, analyzing, and visualizing log data is an important aspect of understanding distributed systems. CloudWatch provide an excellent support for searching, analyzing, and visualizing log instantaneously. Alternatively, one can also use Amazon Elastic Search (ES) together with Kibana. Amazon ES can be used for full-text search, structured search, analytics, and all three in combination. Kibana is an open source data visualization plugin for Amazon ES that seamlessly integrates with it.



Another alternative solution for log analysis can be – Amazon Redshift together with Quick Sight. It's also possible to stream logs into Amazon Redshift through Amazon Kinesis Data Firehose.



Chattiness

Due to the nature of the microservices, there are multiple component interacting with multiple other components. To keep the component interaction effective REST over HTTP/S is preferred due to lightweight nature, however extremely large volume can be problem. Implementing right catching can reduce the chattiness and latency, however a right balance needs to be set in order to balance the good caching rate and timeless/consistency of the data. AWS provided multiple caching option through AWS Elastic Cache; API gateway also provides in build cache layer to reduce the load on the backend service.

Auditing

Due to nature of the microservice, it difficult to track all users' changes performed on each of the microservices. AWS Cloud Trail provided an auditing action on the services, any actions performed by the users are recorded on S3 bucket which can be later evaluated for auditing. Multiple microservices running on different AWS account can be configured to aggregate CloudTrail logs into a single S3 bucket for consolidated auditing.

Resource Inventory and Change Management

It important to monitor the changes and ensure that the changes are not violating the organization policies, in case of any policy violation appropriate alert needs to be raised to the concern teams to the address the event. For Resource Inventory and change management AWS Cloud Config provide an appropriate functionality which can be leveraged to track and mange infrastructure changes and immediately (automatically) react to the non-compliant changes.

Conclusion

Microservices architecture is a distributed design approach intended to overcome the limitations of traditional monolithic architectures. Microservices help to scale applications and organizations while improving cycle times. However, they also come with a couple of challenges that might add additional architectural complexity and operational burden.

AWS offers a large portfolio of managed services that can help product teams build microservices architectures and minimize architectural and operational complexity. This whitepaper guides you through the relevant AWS services and how to implement typical patterns, such as service discovery or event sourcing, natively with AWS services

AWS Best Practices – DDoS

https://d0.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf

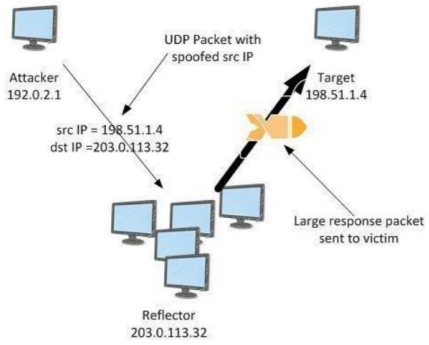
- DoS – Denial of Service
- DDoS – Distributed Denial of Service

DDoS can be targeted to Infrastructure layer or at the application layer.

Infrastructure Layer Attacks.

UDP reflection attacks – In case of UDP reflection attack, the attacker spoof the UDP packets by changing srcIP of the packet with the target IP, and send the request to the multiple systems (Reflector/mediator) , on receiving the

requests the system will send their response to the srcIP with amplification of request of the UDP packet(s) which is the target's IP address.



TCP SYN Flooding attacks In a SYN flood attack, a malicious client sends a large number of SYN packets, but never sends the final ACK packets to complete the handshakes. The server is left waiting for a response to the half-open TCP connections and eventually runs out of capacity to accept new TCP connections.

WordPress XML-RPC attack (also known as WordPress pingback flood) – in this attack attacker misused XML-RPC function of the sites hosted on WordPress, the XML-RPC function of WordPress sites helps sites to ping back another site hosted on WordPress once link is generated.

Application Layer Attack:

HTTP Flood Attack: In this types of attack specific HTTP request is targeted OR in case of a more complex HTTP Flooding attack, it emulates a human interaction pattern with the application with an intend to outrun the server resource capacity. These attacks are difficult to track using request limiting.

Cache Busting attack: In this type of attack, a type of HTTP flood that force CDN not to cache objects, rather reach out to the origin server for each request overloading the origin server. This is achieved by creating variation in the query string.

DNS Query Flood Attacks: In this case the DNS queries are altered in a way that DNS need to forward the request to authoritative servers to resolve the DNS quires, in order to bring down or overload the DNS service.

TLS negotiation Attack web application is delivered over TLS, an attacker can also choose to attack the TLS negotiation process. TLS is computationally expensive so an attacker can reduce a server's availability by sending unintelligible data.

Mitigating DDoS

AWS default protect DDoS are followed free of charges. AWS offer multiple layer protection for DDoS attack. User can also use other AWS service to reduce the risk of DDoS attack.

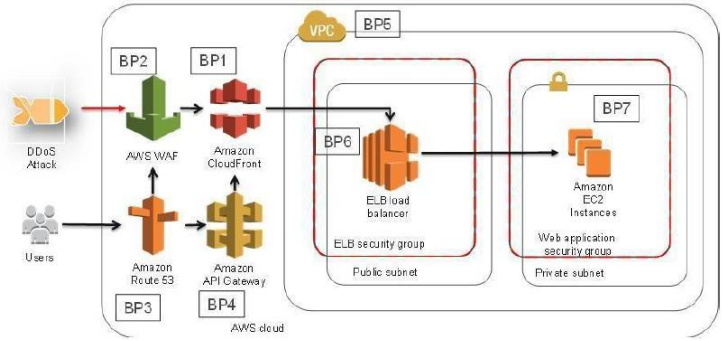
Different techniques to minimized the DDoS attack

- Minimized the surface area
- Learn the normal behaviour
- Be ready to scale
- Safeguard expensive resources that are hard to scale.
- Have a plan to deal with DDoS

Additionally, you can leverage AWS services that operate from edge locations, like Amazon CloudFront and Amazon Route 53, to build comprehensive availability protection against all known infrastructure layer attacks. IT helped in improving DDoS resilience of the application when the serve web application traffic from edge locations distributed around the world.

Several specific benefits of using Amazon CloudFront and Amazon Route 53 include the following:

- AWS Shield DDoS mitigation systems that are integrated with AWS edge services, reducing time-to-mitigate from minutes to sub-second.
- Stateless SYN Flood mitigation techniques that proxy and verify incoming connections before passing them to the protected service.
- Automatic traffic engineering systems that can disperse or isolate the impact of large volumetric DDoS attacks.
- Application layer defense when combined with AWS WAF that does not require changing your current application architecture (for example, in an AWS Region or on-premises datacenter).



Reference architecture of DDoS

	AWS Edge Locations		AWS Regions			
	Amazon CloudFront (BP1) with AWS WAF (BP2)	Amazon Route 53 (BP3)	Elastic Load Balancing (BP6)	Amazon API Gateway (BP4)	Amazon VPC (BP5)	Amazon EC2 with Auto Scaling (BP7)
Layer 3 (for example, UDP reflection) attack mitigation	✓	✓	✓	✓	✓	✓
Layer 4 (for example, SYN flood) attack mitigation	✓	✓	✓	✓		
Layer 6 (for example, TLS) attack mitigation	✓		✓	✓		
Reduce attack surface	✓	✓	✓	✓	✓	
Scale to absorb application layer traffic	✓	✓	✓	✓	✓	
Layer 7 (application layer) attack mitigation	✓	✓	✓ (if used with AWS WAF)			
Geographic isolation and dispersion of excess traffic, and larger DDoS attacks	✓	✓				

AWS Shield Advance can be configure with CloudFront and Route53 , and for selected region its also available for protecting Classic LoadBalancer, Application LoadBalancer, ElasticIP , and for Network LoadBalancer.

With AWS Shield Advanced, you get the following additional benefits:

- Access to the AWS DDoS Response Team (DRT) for assistance in mitigating DDoS attacks that impact application availability.
- DDoS attack visibility by using the AWS Management Console, API, and Amazon CloudWatch metrics and alarms.
- Access to the Global Threat Environment dashboard, which provides an overview of DDoS attacks observed and mitigated by AWS.
- Access to AWS WAF, at no additional cost, for the mitigation of application layer DDoS attacks (when used with Amazon CloudFront or ALB).

- Automatic baselining of web traffic attributes, when used with AWS WAF.
- Access to AWS Firewall Manager, at no additional cost, for automated policy enforcement. This service lets security administrators centrally control and manage AWS WAF rules.
- Sensitive detection thresholds which routes traffic into DDoS mitigation system earlier and can improve time-to-mitigate attacks against Amazon EC2 or NLB, when used with EIP.
- Cost protection that allows you to request a limited refund of scaling-related costs that result from a DDoS attack.
- Enhanced service level agreement that is specific to AWS Shield Advanced customers.

For web applications, one can use ALB to route traffic based on its content and accept only well-formed web requests. This means that many common DDoS attacks, like SYN floods or UDP reflection attacks, will be blocked by ALB, protecting your application from the attack. When ALB detects these types of attacks, it automatically scales to absorb the additional traffic.

For NLB any traffic received for a valid listener will NOT be absorbed, other traffic will be absorbed.

Amazon CloudFront only accepts well-formed connections, which helps prevent many common DDoS attacks, like SYN floods and UDP reflection attacks, from reaching your origin. DDoS attacks are also geographically isolated close to the source which prevents the traffic from impacting other locations.

Amazon Route 53 uses techniques like shuffle sharding and anycast striping, that can help users access your application even if the DNS service is targeted by a DDoS attack.

By using AWS WAF, you can configure web access control lists (Web ACLs) on your CloudFront distributions or Application Load Balancers to filter and block requests based on request signatures. Each Web ACL consists of rules that you can configure to string match or regex match one or more request attributes, such as the URI, query string, HTTP method, or header key. In addition, by using AWS WAF's rate-based rules, you can automatically block the IP addresses of bad actors when requests matching a rule exceed a threshold that you define. Requests from offending client IP addresses will receive 403 Forbidden error responses and will remained blocked until request rates drop below the threshold. This is useful for mitigating HTTP flood attacks that are disguised as regular web traffic.

To block attacks from known bad acting IP addresses, you can create rules using IP match conditions or use Managed Rules for AWS WAF offered by sellers in the AWS Marketplace that will block specific malicious IP addresses that are included in IP reputation lists.

To help identify malicious requests, review your web server logs or use AWS WAF's logging and Sampled Requests features. With AWS WAF logging, get detailed information about traffic that is analyzed by your Web ACL.

Using AWS Firewall Manager can help simplify managing AWS WAF rules across your organization. By using AWS Firewall Manager, you can enable AWS WAF across many accounts and resources, including creating rules that are automatically applied to existing or new accounts in your organization.

Reduce Attack Surface

If one is subscribed to AWS Shield Advanced, then their register Elastic IPs (EIPs) as Protected Resources. DDoS attacks against EIPs that have been registered as Protected Resources are detected more quickly, which can result in a faster time to mitigate. When an attack is detected, the DDoS mitigation systems read the NACL that corresponds to the targeted EIP and enforce it at the AWS network border. This significantly reduces your risk of impact from a number of infrastructure layer DDoS attacks.

If one is using Amazon CloudFront with an origin that is inside of your VPC, you should use an AWS Lambda function to automatically update your security group rules to allow only Amazon CloudFront traffic. This improves your origin's security by helping to ensure that malicious users cannot bypass Amazon CloudFront and AWS WAF when accessing your web application.

With Edge-to-Origin Request Headers, one can add or override the value of existing request headers when Amazon CloudFront forwards requests to your origin. You can use the *X-Shared-Secret* header to help validate that requests made to your origin were sent from Amazon CloudFront.

When you use Amazon API Gateway, you can choose from two types of API endpoints. The first is the default option: edge optimized API endpoints that are accessed through an Amazon CloudFront distribution. The distribution is created and managed by API Gateway, however, so you don't have control over it. The second option is to use a regional API endpoint that is accessed from the same AWS region in which your REST API is deployed. We recommend that you use the second type of endpoint, and then associate it with your own Amazon CloudFront distribution. By doing this, you have control over the Amazon CloudFront distribution and the ability to use AWS WAF for application layer protection.

When you use Amazon CloudFront and AWS WAF with Amazon API Gateway, configure the following options:

- Configure the cache behavior for your distributions to forward all headers to the API Gateway regional endpoint. By doing this, CloudFront will treat the content as dynamic and skip caching the content.
- Protect your API Gateway against direct access by configuring the distribution to include the origin custom header x-api-key, by setting the API key value in API Gateway.
- Protect your backend from excess traffic by configuring standard or burst rate limits for each method in your REST APIs.

Operation Techniques

Use CloudWatch to create different metric to have an better observability on the DDoS attack monitoring. Also, use VPC Flowlogs that can give a clear idea about the normal and abnormalities seeing due to DDoS attack.

AWS Best Practices - Cloud Data Migration*

https://d1.awsstatic.com/whitepapers/Storage/An_Overview_of_AWS_Cloud_Data_Migration_Services.pdf

There are different services to transfer data to/from on premises to AWS cloud. Each services offers different solution and various levels of speed, security, cost and performance.

Challenges related to cloud migration

Security / Data Sensitivity:

AWS Direct Connect: Secure dedicated line / IAM controlled access to the Direct Connect console/ Can be integrated with AWS Cloud trail to capture all the API calls made by the customer or on behalf of the customer.

AWS Import/Export Snow: IAM access control for accessing AWS Import/Export console. Integrated with KMS encryption for providing data encryption at rest. Use industry standard TPM (Trusted Platform Module) to prevent an unauthorized modification of the hardware, firmware, or the software to physically secure the AWS Snowball device.

Storage Gateway: Storage Gateway console secure by IAM access, Data transmitted through storage gateway are encrypted using SSL/TSL, Data stored withing the Storage gateway are encrypted using AES 256 encryption. Authentication between a storage gateways and iSCSI initiator can be authenticated using Challenge-Handshake-Authentication Protocol (CHAP).

Amazon S3 Transfer Accelerator: IAM can be used to secure access to S3 bucket, Bucket level/ Object Level access can be granted to individual IAM user or to another AWS account using access policies. Data stored withing S3 bucket can be encrypted using S3-SSE, SSE-KMS, SSE-Customer Key. Or using client-side encryption. Data transmitted to/from S3 are encrypted using SSL/TSL encryption. Additionally, multi-factor authentication can implement to provide additional security.

AWS Kinesis Firehose: IAM is use to restrict access to AWS Kinesis Firehose. Data transmitted through AWS Kinesis Firehose are encrypted using SSL/TSL encryption.

Cloud Migration Tool: (both managed and unmanaged migration tools offered by AWS for lift-and-shift data to/from on premises).

Below formula helps in calculating the number of days needed to transfer data of given size based on the availability of the.

Number of Days = (Total Bytes)/(Megabits per second * 125 * 1000 ** Network Utilization* * 60 seconds * 60 minutes * 24 hours*)*

Page # 6

AWS Best Practices - Migrating AWS Resources to a New AWS Region

Identify all the service running in the current region – identify services that are to be migrated.

Restrict the users within the region – carefully evaluate the IAM policies make sure time base policy are modified to accommodate the new timezone difference.

SSH Key

SSH key are stored within the region – one can associate new key for the new region or can continue to use the same key. Keys are store under ~/.ssh/authorizedKey folder one can copy the public key and reimport the public key to the new instance

Refactor CloudFormation template to use the new keypair or create a new keypair with the same name in the new region.

AMI

Find the matching AMI version in the new region.

EBS

Take snapshot of the of EBS volume and share the same with the new region.

Elastic IP and DNS quires

Change the TTL to lower value before updating the new IP to the A-record.

Reserved instance

Sell reserved instances which are not needed, buy new reserved instance in the new region from AWS marketplace for short term commitments or buy it directly from AWS.

VPC and other network components

VPCs cannot be more from one region to another region, however one can recreate the VPC with same IP range in the new region where they are migrating.

DirectConnect

Need to find a new aws partner for directconnct offering – find new direct connect partner here https://aws.amazon.com/directconnect/partners/.

Leverage DNS to support migration

Use weighted policy to create a route switch for incoming request from source to target destination. Its also possible to gradually migrate the traffic from the source region to destination region.

Migrating S3 bucket

S3 bucket are region specific, thus need to create new S3 bucket in the new region and copy the content into the new region. As S3 bucket have globally unique name, its not possible to create bucket with same name as the source

region, however there is a workaround in case it's important to keep the same name – copy the content into an intermediary bucket then delete the bucket in the source region wait for 24 hours unit the same name is available on the new region. For accessing the bucket using a friendly name – update the CNAME with the new S3 URL, this will help in routing the traffic to the new bucket instead of the old bucket.

One can use AWS console to move the content of the S3 bucket into the new bucket OR can leverage third party tool to achieve the same.

If one is using CLI to copy the content from source bucket to the new destination bucket – use PUT Object - Copy operation, PUT Object - Copy operation performs GET operation and then PUT operation in a single operation.

Migrating S3 -Glacier content

There is no direct way to more the content from one region Glacier storage class to another region storage class, one need to retrieve the content into a temporary S3 location and then uploaded into the new region Glacier. As there is a Max limit of 5TB size for S3 objects, one may have to use byte range while copying the content into S3 to limit the size of the S3 object within 5TB limit.

Migrating EFS

Copy the EFS files to EBS volume if the size is within 16TB and then create & copy the snapshot to the desired region to migrate the files – then create a new EFS copy back the files. Else copy the files to any S3 bucket, migrate the files to the desired location and copy it back.

Migrating Storage Gateway

Repoint the Storage gateway to the new region, backup the content into EFS or S3 and migrate the files to the region. Access the files from the target region S3 bucket.

For Storage Gateway – volume interface: repoint the volume interface to the new region, create a snapshot of the existing EBS volume, copy the snapshot to the new region and then create a new EBS volume from the snapshot and access the EBS volume from the newly configure volume interfaces.

For Storage Gateway – tape interface : retrieve the taps into client location, crate a new tape interface in the client location and copy the content back to the tape interfaces.

Migrating RDS

Database security group – are region specific, then need to be reacted in the new region. Ensure that the correct CIDR blocks are update as per the new region VPC configuration.

For Aurora DB – create a read replica in the target region. Once replicated, promote the read replicate as primary DB.

For other RDS – use data base migration service to migrate the DB to the new region OR Stop the DB , dump all the DB files into a EC2 instance using DB native data extract commands and then copy the files into the desired location RDS instance.

For migrating DynamoDB – use DynamoDB import/export command from the AWS console to copy the content into a S3 location and then re-importing the content back to new DynamoDB instance in the desired instance. NOTE: if the DynamoDB received very high volume of traffic and it cannot be stop for migration – enable DynamoDB streams in the source table and apply them to the destination table.

Migrating SimpleDB <TO BE ADDED – Page# 30 >

Migrating Elastic Cache

For MemeCache – it does not have persistence storage, create a new cluster of the memcache in the target region, let it populate over the time. Data from the old cluster can't be migrated to the new cluster.

For Redis Cache – create a manual backup of the Redis Cluster in the source region and store the same into a S3 bucket. Create a new Redis cluster in the target region and populate the cluster from the data stored on to S3 bucket.

Migrating Redshift

Using cross region snapshot feature create a new snapshot in the target region, restore the cluster in new region from the copied snapshot.

Migrating Athena

Athena should be running on the same region where S3 data is store, recreate the schemas in the new region Athena service and run the query from the new region.

Migrating EMR Cluster

EMR cluster needs to be re-created in the new region, for migrating EMR data are stored in S3 bucket, copy the bucket into the new region and configure the new EMR cluster to use the data from the new bucket.

If the HDFS cluster is used then one can used S3DistCpp command to copy the HDFS data content into S3 bucket and from S3 to target HDFS.

Elastic Search

Elastic Search domain needs to be rebuilt from the exported snapshot of the source domain into target domain in the new region. (exported snapshot from the source domain can be store into S3 for migrating it into the target region).

Amazon SQS

SQS is a regional service – in order to migrate the SQS into target region create a new SQL (FIFO queue) in the target region, reconfigure the application to write messages into the new queue. Also create a separate application to migrate message from the OLD queue into the new queue.

Amazon SNS

SNS is a regional service, one need to create a new SNS topic into the target region and reconfigure the application to the new queue.

Amazon API Gateway

API gateway are regional service, in order to migrate the API gateway into a new region one needs to export the API Gateway definition into swagger definition and then reimport the definition into the API gateway in the desired location.

AWS CloudFormation

One can easily migrate the cloudFormation template to create AWS resources in another region by changing the mapping declaration as per the target region resource reference.

Leveraging CloudFormer tool to replicate resource in the new region. CloudFormer is a template creation tool that enables you to create AWS CloudFormation templates from the pre-existing AWS resources. It can act as starting point, on which one can add their customization.

Migrating API/Custom Scripts

Most of the API invocation have -- region parameters, this needs to according set as per the new region

Remove all SSH keys in old region, this will prevent logging of users into old region instances.

Disaster Recover – Backup & Strategy

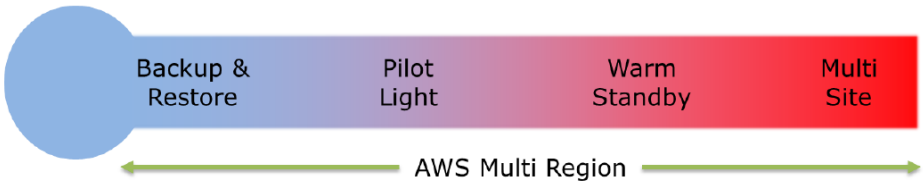
- RTO (Recovery Time Objective): The time it takes after a disruption to restore a business process to its service level, as defined by the operational level agreement (OLA). Time it takes to fully recover from a disaster (system failure)
- RPO (Recovery Point Objective): The acceptable amount of data loss measured in time. Point in time to which the data can be recovered.
- For defining designing backup strategy for backing up and restoring, identity each of the potential failure points and their business impacts, security and regulatory requirements, retention polices, RTOs and RPOs.
- Backup and Recovery strategy needs to be designed for
  - File level recovery
  - Volume level recovery
  - Application level recovery
  - Image level recovery
- Recovery strategy for different scenarios
  - **Cloud Native Infrastructure** where the entire architecture is comprised of AWS services, then the ready built in feature of the AWS can be leveraged for the backup and restoration strategy. In this scenario – for file base storage leverage S3 bucket and for Volume backup one can leverage EBS snapshots stored in S3 bucket (even replicated to another region for high degree of fault tolerance). For cost optimization, incremental snapshots can be created where the first snapshot will have most of the data backed up and remains one will store the incremental changes.

    <u>Volume level Backup</u>
    - Temporally unmounting the disk/volume to ensure consistent backup.
    - Flashing out the buffer memory in case of RAID setup
    - Using agent-based backup solution
    - Creating replica of the Primary volume (while doing this, one need to ensure that single large volume should be sufficient to address the maximum size required).

    <u>Database Backup</u>
    - For backing up of the databases running on the EC2 instances – it can be done by creating backup of the data files using native methods /tools like EBS snapshots
    - For Large Databases which are built on RAID setup – one can offload the performance impact of primary db instance by taking backup from read replicas instead of primary db. The read replicas can have similar RAID configuration as that of the primary db OR can be set to consolidate all RAID volume into a single volume (provided required size of EBS volumes are available).
    - FOR RDS INSTANCES – manual and automated backup are available
    - AUTOMATED BACKUP FOR RDS INSTANCE: this can configure to take full daily backup at the define window set during db instance creation. Using automated backup snapshot in conjunction with the transaction logs, one can recover from any failure up-to 5 min in past. Automated backup can be retained up to 35 days.
    - MANUAL BACKUP (DB snapshot) FOR RDS INSTANCE: this can done to create a point in time backup which can be used to restore the DB or recreate a new DB instance with different endpoint.
  - <u>Image Backup</u>
    - AMI can be created to store image which can be used to quickly provision/restore instance during any failure.

- AWS quota Elastic IP address is 5 Elastic IP addresses per account per region. Elastic IP address are regional constructs – they are confined to a single region ONLY.
- S3 durability 99.9999999999 11 9's
- Essentials infrastructure components that needs to be consider for Backup & Recovery Strategy

- **Region:** Select multiple region, each of which are geographically separate global region, to ensure availability of one or more region during disaster event.
- **Storage:** Durable storage.
    - **S3 bucket:** Store content redundantly over multiple availability zone with high durability of 99.9999999999 11 9's. MFA & versioning to avoid accidental delete. On premises data can be backed up into S3 through – Direct Connection, through AWS import/export portable devices, through internet. On an event of DR uploaded data can be quickly retrieved or can be used to mapped to a new instance.
    - **Glacier:** For archival storage.
    - **EBS volume:** Point in time snapshot can be created, independent of the instance lifecycle which can be stored in S3 to achieve high durability of the EC2 instance data.
    - **AWS import/export:** this feature can be used to import/export large amount of data to/from AWS using portable devices.
    - **AWS storage gateway:** This can be used to automatically backup on-premises data into AWS cloud using iSCSI protocol. One of the following variants can be used based on the need
        - **AWS Storage Gateway –** cached volumes
        - **AWS Storage Gateway –** store volumes
        - **AWS Storage Gateway –** Tape volumes
- **Compute:** Durable compute
    - **AWS EC2 instance:** compute instance can be created from store AMI (AWS machine image)
    - **AWS VM import/export:** this can be used to migrate & run on premises VM into AWS.
- **Networking:** In an event of disaster, network settings can be changed quickly to route production traffic from failed instance to DR instances. The following AWS components can help to quickly switch production traffic.
    - **Route53:** Highly available, AWS managed DNS service.
    - **Elastic IP address** – it can programmatically remap from the failed instance to new instance. Software licenses which are allocated to the MAC address , instead of remapping the elastic IP address Elastic NIC can be remapped.
    - **Elastic Load balancer** – it can automatically route the production traffic to healthy instance within the region. (*it's a regional service, can't route traffic outside the region*).
    - **AWS VPC:** Fully controllable data center within AWS region**.**
    - **AWS Direct Connect:**
- Database:
    - **AWS RDS –** supports, manual and automatic snapshots, multi-AZ configuration to prevent/recover from the failure.
    - **Dynamo DB –** fully managed service from AWS
    - **Redshift –** support manual and automatic snapshot which can be use to restore the failed instance within same or different AWS region.
- **Deployment Orchestration:** On an event of any failure, deployment orchestration quickly helps in restore failed instances using deployment automation and post startup installation and configuration.
    - **AWS Cloudformation:** Enable to provision AWS resources in orderly and predictable fashion. Entire configuration can be stored in a single file which can also be versioned.
    - **AWS Elastic BeanStalk:** Easy way to deploy and scale application on AWS, it's a fully managed service on an event of any failure detection elastic beanstalk replace the underline services with healthy instance.
    - **AWS OpsWorks –** Chef base infrastructure as a code service which can be used in conjunction with AWS CloudFormation to automatically provision new stack in an event of an failure and replace the host in the newly created stack.
- Different Type of Disaster Recovery Strategy



- **Backup & Restore:** There is NO running DR site, primary site data are backed up on a remote site (*un-affected by the primary side failure*). On an event of a failure the backed-up data will be used for restoring the instances.

Preparation Phase	Recovery Phase

- **Pilot Light:** Minimum version of the environment is always running on the DR site. On an even of any failure, remain system can be recreated with ease and primary site can be restored.

Preparation Phase	Recovery Phase

- **Warm Standby:** A scale down version of the environment will be running on the DR site. In an event of any failure the DR system will be scale out to cater production traffic.

Preparation Phase -	Recovery Phase
1. Set up EC2 instances to mirror data 2. Create and maintain AMI for scaling up of the new instances. 3. Run AWS resources with minimal footprint. 4. Patch and update stand-by instance to keep in sync with the production environment.	1. Scale up EC2 instance (horizontal scaling) or use a large EC2 instance (vertical scaling). 2. Manually change the DNS entry in the Route53 or use Route53 health check to route traffic to the new instance(s). 3. Scale DB layer to handle the scale up load.

- **Active-Active:** In case of multi-site active-active configuration, fully functional production ready instances will be running in parallel on the DR site. On an event of a failure the production traffic will be route to the alternative site. **Lowest RTO, highest DR maintenance cost.**

Preparation Phase	Recovery Phase
1. Set the DR site equivalent to the production site. 2. Set the Route53 weight to send no/small traffic to the DR site while sending full/major traffic to the	1. Manually change-over to the DR site 2. Use autoscaling to right size the instance fleets

- Key factors to be consider while replicating the data
  - Distance between the sites – larger the distance more latency and jitter
  - Availability of the bandwidth –
  - Data rate required by the application – should be lower than the available bandwidth.
  - Replication technology – Synchronous Replication / Asynchronous Replication Should be done in parallel to optimize replication.
- Fallback Strategy – Once the primary site is back, the traffic needs to be routed back to the primary site from DR site. Based on the different DR strategy the step will differ

For Backup and restore Strategy	For Pilot light / Warm Standby / Active-Active
1. Freeze the data change on the DR site. 2. Take backup of the DR site 3. Restore Primary site using the DR site 4. Unfreeze the data change on the DR site, ensure DR site is getting backed up by the primary site data.	1. Use reverse mirror of the data from the DR site to the primary site. 2. One Primary site is restored completely. 3. Route the traffic back to the primary site

- Disaster readiness
  - Testing – schedule regular game-days. Ensure that game-day scenario is aligned to the real disaster scenarios.
  - Effective Monitoring and Alert

- Keep Backup always on.
- Evaluate licensing agreement – ensure that correct licensing option is selected.

AWS Security Incident Reporting Guide

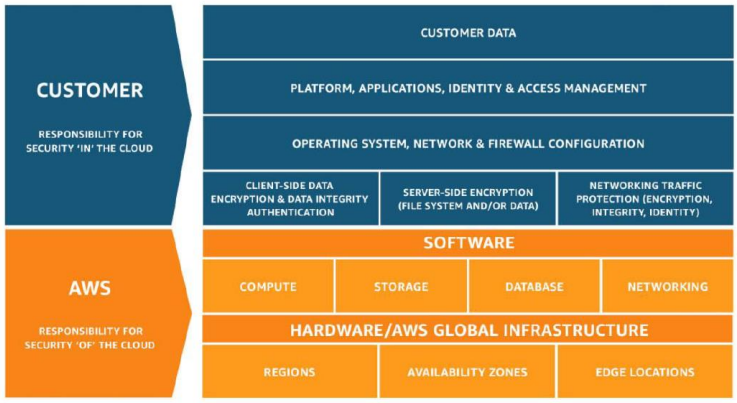
From AWS CAF (Cloud Adaptation Framework) Security prospective it consists of following four components:

- **Directive Control**: Established governance, risk and compliance model within the working environment.
- **Preventive Control**: Protect workload and mitigate threat and vulnerability.
- **Detective Control**: Provide visibility and transparency over the operations.
- **Responsive Control**: Drive remediation of potential deviation from the security baseline.

Foundation of Incident reporting are **EDUCATE**, **PREPARE**, **SIMULATE** and **ITERATE**.

EDUCATE

Share responsibility model.



Design goals for cloud response

- Establish response Objective
- Response using cloud
- Know what you have and what you need
- Use redeployment mechanism
- Automate where possible
- Chose scalable solution
- Learn and improve your process

Cloud Security Incident domains

There are three domains within customer responsibility where security incident can happen – ***service domain, infrastructure domain*** and ***application domain***.

Indicator of cloud security events

- Log & Monitoring
- Billing Activity
- Threat Intelligence
- Partner Tool – AWS Partner Network (APN) – Security Partner Solution, Security Solution in AWS Marketplace.
- AWS Outreach – AWS response to Abuse and Compromise sections.
- One-Time contact: there should be a well define / ticketing solution available which employees can use to reach out to the security when they see any abnormality related to security.

Understanding Cloud Capability: Understand the various services that AWS offers to detect/response to a security incident.

Data Privacy: Even AWS can't see customer data.

AWS Response to Abuse and Compromised: AWS team proactively monitors AWS account for any suspicious and malicious activities, on such event they report and shutdowns un authorized activities running on AWS. Majority of the abuse can be categorised as

- **Compromised source**: an unpatched EC2 instance which can infect and become a botnet agent.
- **Unintentional Abuse**: Overly aggressively web crawling, this might be categorized as denial-of-services by some websites.
- **Secondary Abuse**: End users storing malicious files in hosted S3 bucket
- **False Complain**: Many a time, internet users may report a normal activity as abusive activities.

LEARN.

Automate the process such that, it gives humans has more time to focus and increasing security measure and spend time in corelating events, practicing simulations, device new response procedure, perform research, develop new skills, test and build new tools.

- Define Role and responsibilities
- Provide Trainings
- Define Response Mechanism
- Create a Receptive and Adaptive security culture
- Predicting response – partnered with others /share knowledge

IN 1955, Joseph Luft and Harrington Ingham created the Johari Window (window of response), a simple graph to represent the knowledge of the partner and the internal tribe. Through it was not intend to be use for security risk, the same can be easily extend for security risk.

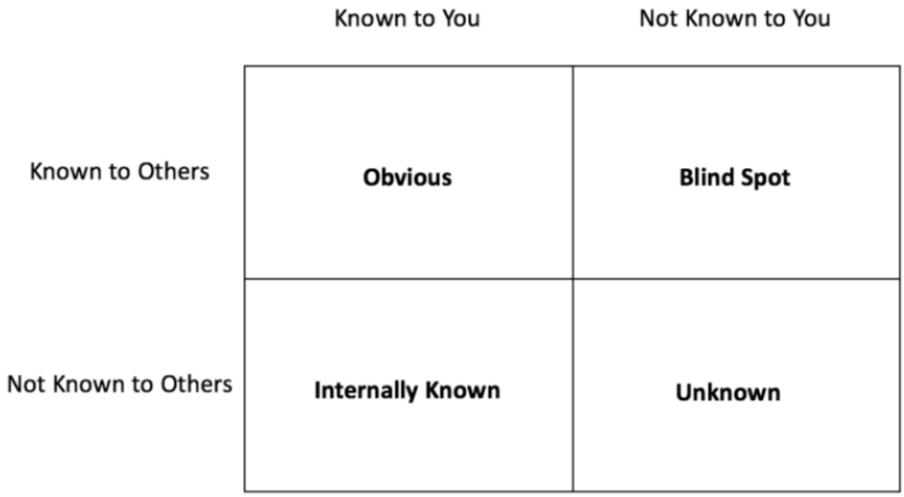


Figure 2 – Window of Response

For **Unknown** quarter, one can follow the below method to reduce the security risk.

Defence security assertion: Define security assertion, make it easily searchable. Start from early cloud days, then starting it late.

Education, communication and research: Create a cloud security expert in your team or leverage external experts, to scrutinized your environment. Create a feedback look, between the security expert and the engineering team.

Reduce Attack Surface: Improve defect to reduce the attack surface for unknown threats.

Threat Intelligence: Subscribe to continuous feed of current and relevant security threat, risk and indicators from around the world.

Alerts: Generate notification alert for all unusual malicious activities.

Machine Learning: Leverage Machine Learning to find complex abnormalities of the organization or specific persona. AWS Macie and AWS Gard Duty can be used for the same. Extend the business-centric data-lakes architecture to create security-centric data lakes, store all kinds of security information into security data lake, and leverage AWS services to derived patter from complex abnormalities.

PREPARE.

Prepare access to AWS Account: ensure that security response team have access to the environment where security incident was reported. Place a mechanism in place for the security teams to get access quickly. As its common practice to have multiple AWS account linked to a master account (payee account) it might be required security team to have cross account access - ensure that security team have cross account access. This can be leverage using ***service control policies***.

In-direct access: Security team, assist the application team/account owner to apply remediation on an event of incident.

Direct access: Application team/account owner deploys IAM roles for the security teams to assume such roles during incident event for applying remediation.

Alternative Access: Security incident responder can login into a secluded/secure account to investigate and remediate threat instead of having direct or indirect access to the actual environment.

Automated Access: Instead of provisioning access for the incident responder, create role specifically for the automation resources like (EC2 instance and Lambda). When incident occurs automation resources can assume such roles and act on implementing remediation. One can use AWS System Manager Run Command to run administrative tasks remotely & securely on any EC2 instance where AWS System Manager Agent is installed.

Managed Service Access: AWS account managed by the trusted third-party partners, to manage/implement/remediate on an event of an incident.

Prepare Process: Once the access is provisioned, there should be clear process defined which security incident reporting team can follow in order to investigate and remediate an incident.

Decision Tree: Sometimes, different actions need to be implemented based on the incident event.

Use Alternative Account: Security incident remediation team, may need to investigate the threat in a separate isolated account. AWS Organization can be used to create a separate forensic environment to analysed the threats. Auto infrastructure automation to create investigating environment mimicking the actual environment in the alternative account.

View or Copy Data: Security responders should have view access to the security logs. Appropriate IAM permission should be in place for the responders to copy point-in-time logs into investigation S3 bucket from the production bucket, in order to analyse the incident. Data can be store in S3 storage or can be archive under S3 Glacier for long term retention. One can also protect the data using S3 Glacier Vault Lock, where one can easily apply compliance base rule for long term retention.

Share EBS Snapshot for incident investigation – if the snapshot to encrypted make sure cross account access to the CMK (customer managed key) is provided along with the permission for copying the EBS snapshot.

CloudWatch Logs and VPC flow logs are store centrally. User Kinesis to process the logs from different AWS account into a single AWS account. While storing the data ensure the storage is immutable to protect the data integrity.

Launch Resource Near the event: Incident occurred on premises can also be investigate on the cloud environment, there are better accessibility to service to investigate & response to an incident. It may be beneficial to have a long term separate isolated AWS account for investigation, and for long term storage and legal usages.

Isolated Resources: There can be a need to create an isolate the resources (system) to perform forensic investigation. Best practice for launching a forensic investigation instance, Create AMIs and store the AMI or

CloudFormation template so that it can be quickly provision when needed, this will also helps in standardising the forensic workstations

1. Chose relevant AMI (windows or LUNIX), for launching forensic investigation workstation.
2. Launch EC2 instance based on the AMI
3. Harden the OS, remove un wanted software packages – configure relevant auditing and Logging mechanism.
4. Install open source / private toolkit software required for investigation
5. Stop EC2 instance, create a AMI from the EC2 instance once its stopped.
6. Weekly/monthly build EC2 instance from the AMI and apply patches to the installed software/platforms.

Cloud Provide Support

- **AWS Support** – best practices document, whitepaper, AWS documentations, support forums etc. Chose appropriate support plans.
- **DDoS Support** (Denial of Service OR Distributed Denial of service). AWS provided AWS Shield for DoS (Standard or Advance) support. DoS Standard is free for all customers which includes standards known technique – comprehensive availability protection against well-known infrastructure attacks. User also enrol for advance DDoS protection.

Simulate

Security Incident response simulation (SIRS) helps in identifying

1. **Validate Readiness**
2. **Develop Confidence – learn from simulation and training staffs.**
3. **Follow Compliance and contractual obligations**
4. **Being agile – incremental improvement with leaser focus.**
5. **Become faster and improving tools**
6. **Refine communications and accelerations**
7. **Developing comfort with rare and unexpected scenarios**

Simulation Steps:

Simulated examples:

Iterate

Create a feedback loop, to know what is working AND what is not working. One can create new procedure or update the existing procedure based on the feedback received.

Runbook

Runbook is an organization procedure, which consist of a task or series of tasks which need to be refer when there is an incident occurred. Keep re iterating the tasks to improve the core logic.

Automation

Once the core logic is defined in the runbook, one should look forward towards the automate the task(s). For automation comprehensive AWS APIs can be used. Automate the security engineering and operation functions this using comprehensive AWS API and tool, one can improve the automated the manual processes by programming automating steps in the process.

Event Driven Response: With an event-driven response system, a detective mechanism triggers a responsive mechanism to automatically remediate the event. To create this event-driven architecture, you can use AWS Lambda, which is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources on your behalf. Ideally, the goal of event-driven response is for the Lambda responder function to perform the response tasks and then notify the responder that the anomaly has been successfully resolved with any pertinent contextual information.

Incident Response Example

- Don't use IAM root account for day to day activity.
- Use IAM policy to prevent abuse on critical service.
- Use AWS Trusted Advisor to provide number of checks – created with AWS Partner Competency program.

Infrastructure Domain incident.

For EC2 instance incident management

- Capture – capture Ec2 metadata
- Protect – protect acedental termination
- Isolate – isolate the instance
- Deatch – detached the instance from all autoscaling group
- Deregister – deregister the instance from ELB
- Snapshot – snapshot the ebs volume for further investigation.
- Tag – add relevant tags to the EC2 instance while it's in quarantine. For example – AWS System Manger can automatically capture the required information and lambda function can isolate the instance if the instance is tag for isolation.

Overview of Deployment Options on AWS*

Whitepaper link : https://d0.awsstatic.com/whitepapers/overview-of-deployment-options-on-aws.pdf

Practicing Continuous Integration and Continuous Delivery on AWS

Whitepaper link : https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf

There is a general misconception of Continuous Delivery mean continuous deployment to production. Continuous delivery does not mean continuous production deployment, it only mean continuous delivery to staging environment, beyond that business team will be involved in making decision to deploy the changes to production as per the business requirement through a seamless painless approval process.

Benefit of CI/CD

- Automated software release
- Improve developer productivity
- Improve code quality
- Deliver faster

Number and the nomenclature of the staging environment are NOT fixed based on busines needs this can be change to fix the needs.

CI/CD is a journey, it's a continuous process where constant feedback from the end-use needs to re-feed to make the process better.

A mature CI/CD process should have more of staging environment specified performance and compliance, security and UI test included as part of the delivery pipeline. It should have integration with other systems like code review, issue tracking and event notification etc. It should have ability to take care of the DB schema changes. Additionally, it should also have steps for auditing and business approval embedded in it.

AWS provide many CI/CD tool like – codeStart, codeDeploy, CodeCommit, CodePipeline.

Deployment process

- AllAtOnce – in place deployment, everything in one go.
- Rolling – Canary deployment where a portion of server instances are upgraded with new code change, as the code passes through the test, more and more servers are added to this. Other variant of the rolling deployment strategy are oneAtATime and HalfAtAtime.
- Immutable – blue green deployment where a new code added to the new instance and a small amount of traffic is routed to the new code instances (GREEN INSTANCE) while majority of the code still server by old instances (BLUE ISNTACNE) as the code pass through tests, traffic to GREEN INSTANCE are increase in an incremental fashion.

Database changes tool like LiquidBase and Flyway can track and migrate DB schema changes.

Do:

- Treat your infrastructure as code
    - Use version control for your infrastructure code.
    - Make use of bug tracking/ticketing systems.
    - Have peers review changes before applying them.
    - Establish infrastructure code patterns/designs.
    - Test infrastructure changes like code changes.
- Put developers into integrated teams of no more than 12 self-sustaining members.
- Have all developers commit code to the main trunk frequently, with no long-running feature branches.
- Consistently adopt a build system such as Maven or Gradle across your organization and standardize builds.
- Have developers build unit tests toward 100% coverage of the code base.
- Ensure that unit tests are 70% of the overall testing in duration, number, and scope.
- Ensure that unit tests are up-to-date and not neglected. Unit test failures should be fixed, not bypassed.
- Treat your continuous delivery configuration as code.
- Establish role-based security controls (that is, who can do what and when).
- Monitor/track every resource possible.
- Alert on services, availability, and response times.
- Capture, learn, and improve.
- Share access with everyone on the team.
- Plan metrics and monitoring into the lifecycle.
- Keep and track standard metrics.
    - Number of builds.
    - Number of deployments.
    - Average time for changes to reach production.
    - Average time from first pipeline stage to each stage.
    - Number of changes reaching production.
    - Average build time.
- Use multiple distinct pipelines for each branch and team.

DONT

- Have long-running branches with large complicated merges.
- Have manual tests.
- Have manual approval processes, gates, code reviews, and security
- reviews.

Blue Green Deployment in AWS*

White paper link : https://d0.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf