CloudAcademy

AWS Security Fundamentals

(VPC – IAM – S3)

Table of Content

Author's Introduction

My name is Stuart Scott and I am a Cloud Trainer here at Cloud Academy, specialising in Amazon Web Services (AWS). I have been working in the IT industry for two decades covering a huge range of topic and technologies, from data center and network infrastructure design, to more recently, cloud architecture.

*I am a Certified Data Center Design Professional (CDCDP). My latest achievements are in the Amazon Web Services (AWS) field. I hold the AWS Certified Solutions Architect - Associate certification as well as accreditations as an AWS Business and AWS Technology Professional in TCO and Cloud Economics. This year I was awarded '**Expert of the Year Award 2015**' from Experts Exchange for my knowledge share within cloud services to the community.*

Maintaining a high level of expertise within the IT industry is something any engineer or architect always strives for, and I am no different. As a result learning is an on-going task. Keeping your knowledge up to date and relevant allows you to excel within your role and benefit yourself and your business, leading to a better future.

Because I am unbelievably passionate about cloud security, I authored a series of blogs for Cloud Academy on the subject, which lead me to this project.

I wrote this eBook from the perspective of someone who has had to learn the *fundamentals of AWS and understand the different levels of security that AWS offers. AWS enables me to ensure secure and stringent designs and deployments are possible. I hope this eBook helps those of you who want to do the same.*

Cloud integration and adoption is a strategy that more and more organisations are working hard to implement, and having an understanding of the security of that architecture is essential. In my experience, security is the topic that CEOs and CTOs are the most concerned about. Poor security within the Public cloud can lead to attacks and loss of data, both of which are detrimental to you and your business - to say the least.

Overview of Cloud Security

Now more than ever, the single most significant technology change in the mindsets of organizations specifically for the CTO/CIO is whether or not to adopt Cloud services within their IT environment in some form. More and more industries, ranging from the Financial Sector, Retail and even through to Government, are utilizing Cloud technology in one model or another, whether it be a Private, Public or a Hybrid Cloud configuration.

Adoption and migration to Cloud services for organizations were initially met with high concern because of unknown risks that many leaders were not ready to take. Questions, such as 'is my data safe?', 'who can access my data?", and 'where is my data physically stored?' were continually asked. During this transitional period, people who fully understood the significant advantages Cloud computing could bring if implemented correctly were few and far between. Many advantages of distributed computing were widely publicized so interest built in the face of concern.

Over time, and with the emergence of more readily available training (such as the services offered by cloudacademy.com), the true benefits of shifting to the Cloud were being realized. This shift brought an exponential increase in the number of people who understand how distributed computing works, and the best ways of making use of the Cloud's power.

Utilizing Cloud based services is now regarded as an integral component for many leading companies.

Security has significantly increased and adapted to reassure consumers of Public Cloud offerings, such as Amazon Web Services (AWS). Without this adaption by AWS their customer base would not be in the Millions — as it stands today. They listened to the concerns, worries and questions of their consumers relating to all areas of Security.

They built and implemented controls to put cautious customers at ease. By doing so, the user base of AWS skyrocketed, along with their market share and profits! It is now widely regarded, within the security industry, that the security offered by AWS is more stringent than that of most private data centers.

Cloud Security is now a huge area that businesses must actively address should they wish to harness Cloud services. Failing to have the right security in place can cause tens of thousands of dollars of expense and damage should their environment become maliciously hacked. Financial factors are at risk with cloud security, we understand that. However other critically important business data must be protected from theft to because money can be replaced, but a corporation's reputation can not.

Because of recent high-profile hacks, many larger corporations are investing in Cloud Security experts to focus entirely on ensuring their production environment running within the Cloud is deployed with the protection and restrictions necessary for their workloads. With AWS releasing more services, the need to maintain a solid understanding of the latest security practices is of utmost importance.

Many different roles have appeared within the industry following the needs of Cloud Security experts. These new roles range from Solutions Architects, Security Specialists, Security Consultants, and many other variants that I am sure you will come across -- if you haven't already. Each of these roles varies slightly in their requirements and you can find out more about these roles within a great eBook that covers '[The Cloud Computing Job Market'].

To be a specialist in Cloud Security you must understand the underlying foundations of many of the services that you wish to secure. This allows you to ensure you apply the correct features, tools and concepts at every layer.

On top of this, having a sound understanding of a wide variety of technology stacks, such as Application Development, Network Architecture, Server Infrastructure, and Enterprise Solution Design will allow you to enforce security policies within each, resulting in a robust security strategy across your Cloud environment.

AWS Security Controls and Best Practices

The scope of AWS security is huge, and so within this section I shall cover some of the most common and effective methods of implementing security in your VPC and across the IAM and S3 services.

AWS Shared Responsibility Model

AWS security best practices begin with the AWS Shared Responsibility Model, that dictates which security controls are AWS's responsibility, and which are yours (the user). This model lies at the very foundation of AWS Security.

By the very nature of the phrase – AWS Shared Responsibility Model – we can see that security implementation on the AWS Cloud is not the sole responsibility of any one player, but is shared between AWS and you, the customer. You must decide how you want your resources to sit 'in' the cloud, while AWS guarantees the global security 'of' the Cloud (i.e., the hardware they provide to host and connect your resources).

In my experience, a solid understanding of the AWS Shared Responsibility Model makes it easier to build and maintain a highly secure and reliable environment. Without knowing where I needed to step in and take control of data security, I was never able to properly define just how secure my environment really was.

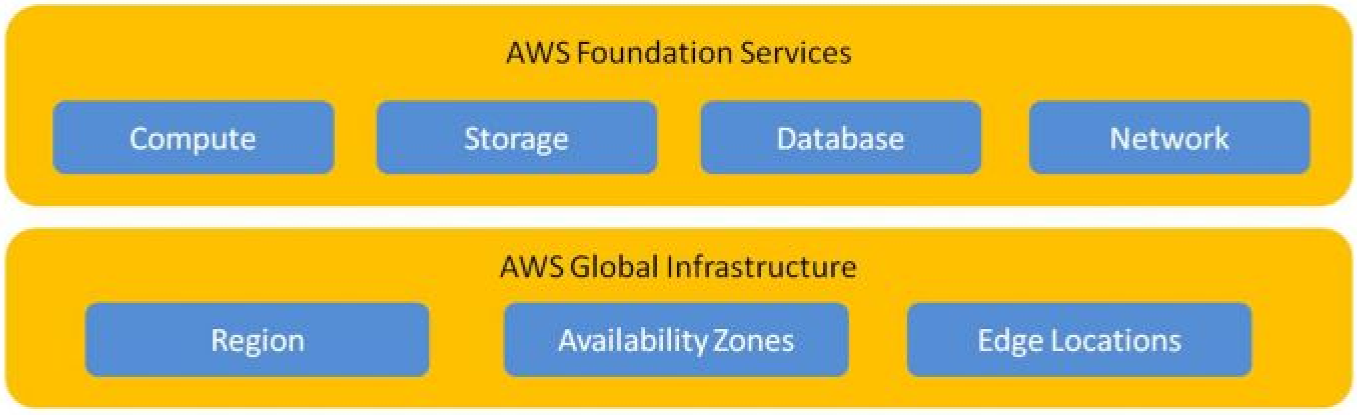
Security is AWS's number one priority in every sense. It's an area into which AWS pours huge capital and energy and near-constant attention.

And there's a reason for this: from speaking with my business contacts in various sectors, it seems security is still one of the main reasons corporations are reluctant to adopt a cloud presence. Overcoming this hesitation requires AWS to be at the very top of security excellence and governance.

Serving over a million customers, AWS's most stringent security standards are already being used for audit purposes by the most security-sensitive customers in the world. Facing so many requirements, AWS is certified and compliant across a huge range of security standards, including PCI DSS, ISO, SOC.

AWS Services are deployed and distributed in exactly the same way throughout their entire global infrastructure. This means a single user accessing a simple S3 bucket for document backups is covered by the same intense security standards as the largest and most demanding corporations and governmental agencies.

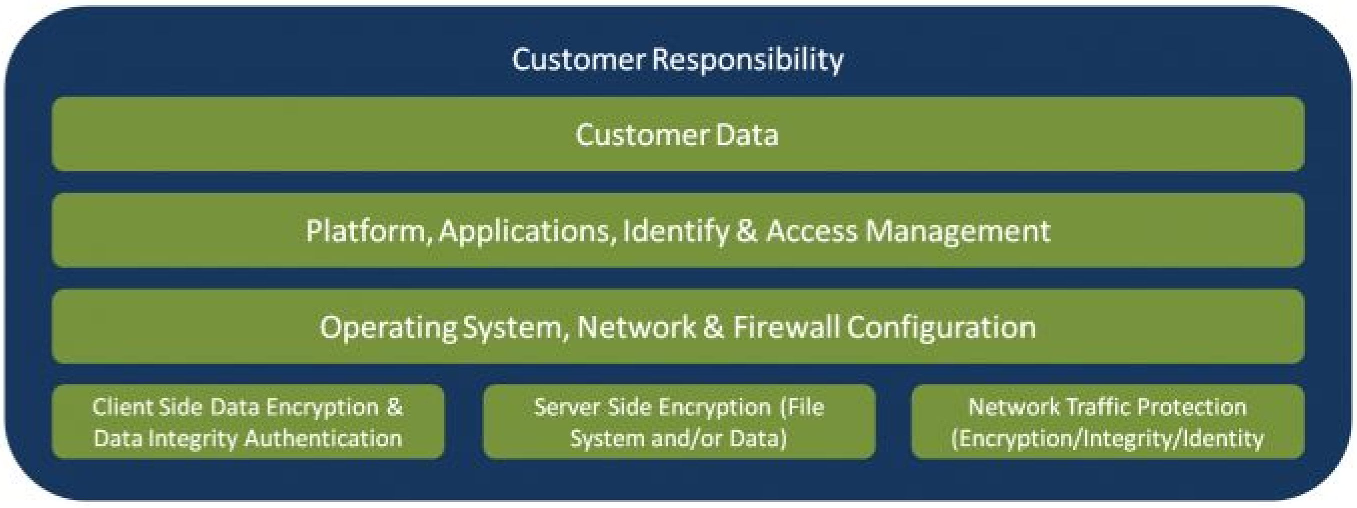
The AWS Shared Responsibility Model: AWS's Job



AWS is responsible for what is known as Security 'of' the Cloud. This covers their global infrastructure elements – Regions, Availability Zones, and Edge Locations, as well as the foundations of their services covering Compute, Storage, Database, and Network.

AWS owns and controls access to their Data Centers where your customer data resides. This covers physical access to all hardware and networking components, and any additional Data Centre facilities including generators, uninterruptible power supply (UPS) systems, power distribution units (PDUs), computer room air conditioning (CRAC) units and fire suppression systems. Some of the security compliance controls mentioned previously are based upon this physical access entry and control. Essentially, AWS is responsible for the components that make up the Cloud, any data put 'into' the cloud then becomes, you guessed it, your responsibility.

The AWS Shared Responsibility Model: Your Job


Customer Responsibility
Customer Data
Platform, Applications, Identify & Access Management
Operating System, Network & Firewall Configuration
Client Side Data Encryption & Data Integrity Authentication
Server Side Encryption (File System and/or Data)
Network Traffic Protection (Encryption/Integrity/Identity

With the basic Cloud infrastructure secured and maintained by AWS, the responsibility for what goes into the Cloud falls on you. From the bottom of the stack, this covers both client and server side encryption and network traffic protection, and then moves up to the security of operating system, network, and firewall configuration, followed by application security and identity and access management.

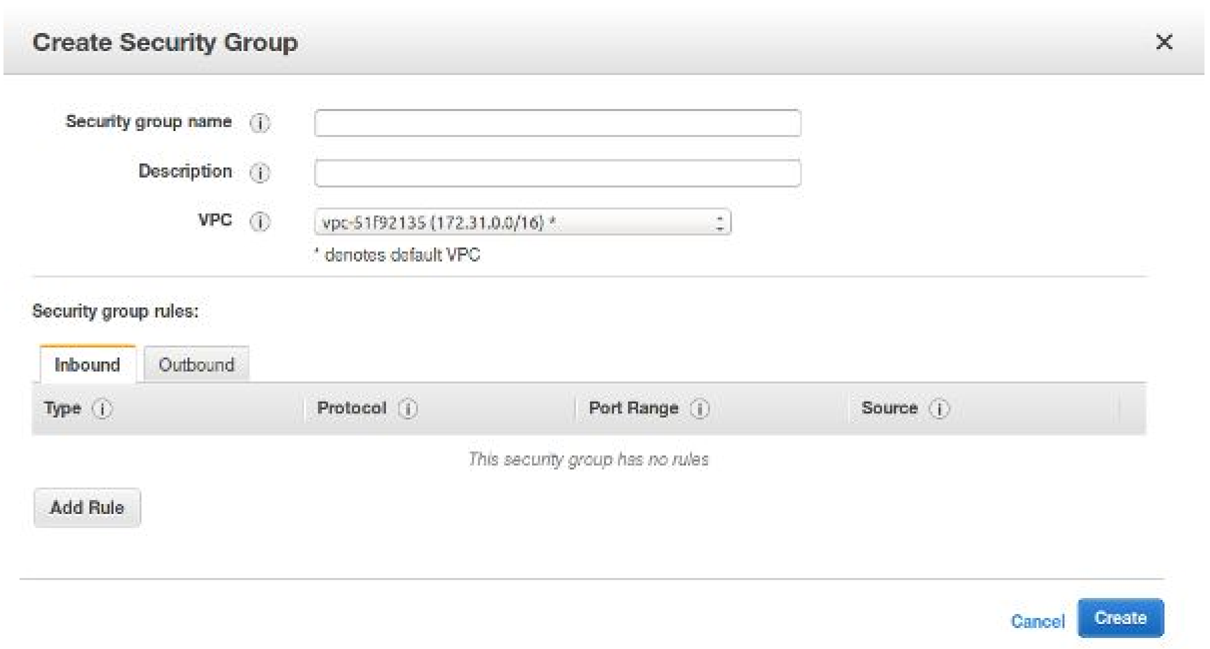
How much of this additional security you wish to implement is entirely your decision. What you choose may depend on the nature of your business or on existing controls you may already have in place. I recommend tightening security just enough to minimize exposure to external threats that could compromise your environment. The important point to remember is that, while AWS provides many powerful security controls, how and when to apply them is not AWS's responsibility.

Instance Level Security

Instance security requires that you fully understand AWS security groups, along with OS patch protocols, key pairs, and your various tenancy options.

Security Groups

AWS security groups (SGs) are associated with EC2 instances and provide security at the protocol and port access level. Each security group – working much the same way as a firewall – contains a set of rules that filter traffic coming into and out of an EC2 instance. There are no 'Deny' rules. Rather, if there is no rule that explicitly permits a particular data packet, it will be dropped.


Create Security Group

Security group name	
Description	
VPC	vpc-51f92135 (172.31.0.0/16) *

* denotes default VPC

Security group rules:

Inbound | Outbound

Type	Protocol	Port Range	Source

This security group has no rules

Add Rule

Cancel Create

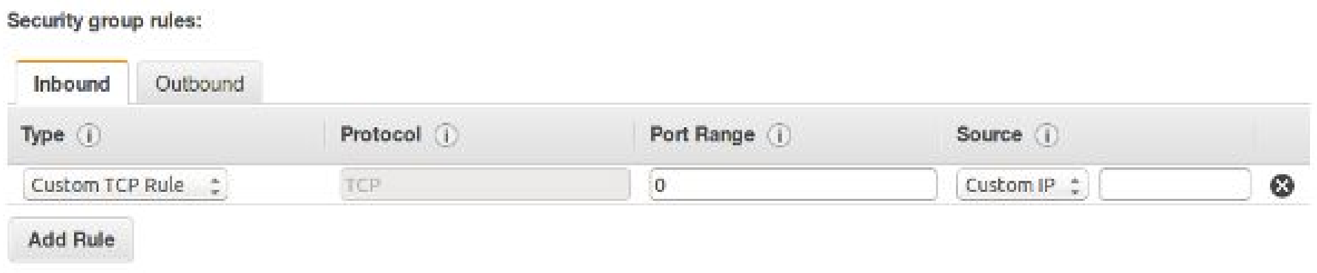
Each security group must have a name, allowing you to easily identify it from account menus. It's always a good idea to choose a descriptive name that will quickly tell you this group's purpose. In fact, you would be well served to define and use a consistent convention for naming all objects in your AWS account.

Security groups exist within individual VPCs. When you create a new group, make sure that it's in the same VPC as the resources it's meant to protect.

AWS Security Groups: Rules

The actual rule set that filters traffic is made up of two tables: 'Inbound' and 'Outbound'. AWS Security groups are "stateful", meaning you do not need the same rules for both outbound traffic and inbound. Therefore any rule that allows traffic into an EC2 instance, will allow responses to pass back out without an explicit rule in the Outbound rule set.

Each rule is comprised of four fields: 'Type', 'Protocol', 'Port Range', and 'Source'. This applies for both 'Inbound' and 'Outbound' rules.



Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	0	Custom IP

Type: The drop down list allows you to select common protocols like SSH, RDP, or HTTP. You can also choose custom protocols.

Protocol: This is typically greyed out, as it's covered by most 'Type' choices. However, if you create a custom rule, you can specify your protocol (TCP/UDP etc.) here.

Port Range: This value will also usually be pre-filled, reflecting the default port or port range for your chosen protocol. However, there might be times when you prefer to use custom ports.

Source: This can be a Network Subnet range, a specific IP address, or another AWS security group. You can also leave access open to the entire Internet using the ‘Anywhere (0.0.0.0/0)’ value.

Security Group Limits

Before building a complex plan that involves creating large numbers of security groups within a single VPC, be aware that you are limited to only 100 security groups per VPC. You can request that AWS increases the limit, but you may notice a network performance impact.

There is also a limit of 250 rules per network interface on your instances. With this in mind, you could create five security groups with 50 rules each, or ten security groups with 25 rules in each. As long as you stay below the overall limit of 250. Also, it’s worth noting that you can’t have more than sixteen security groups per network interface.

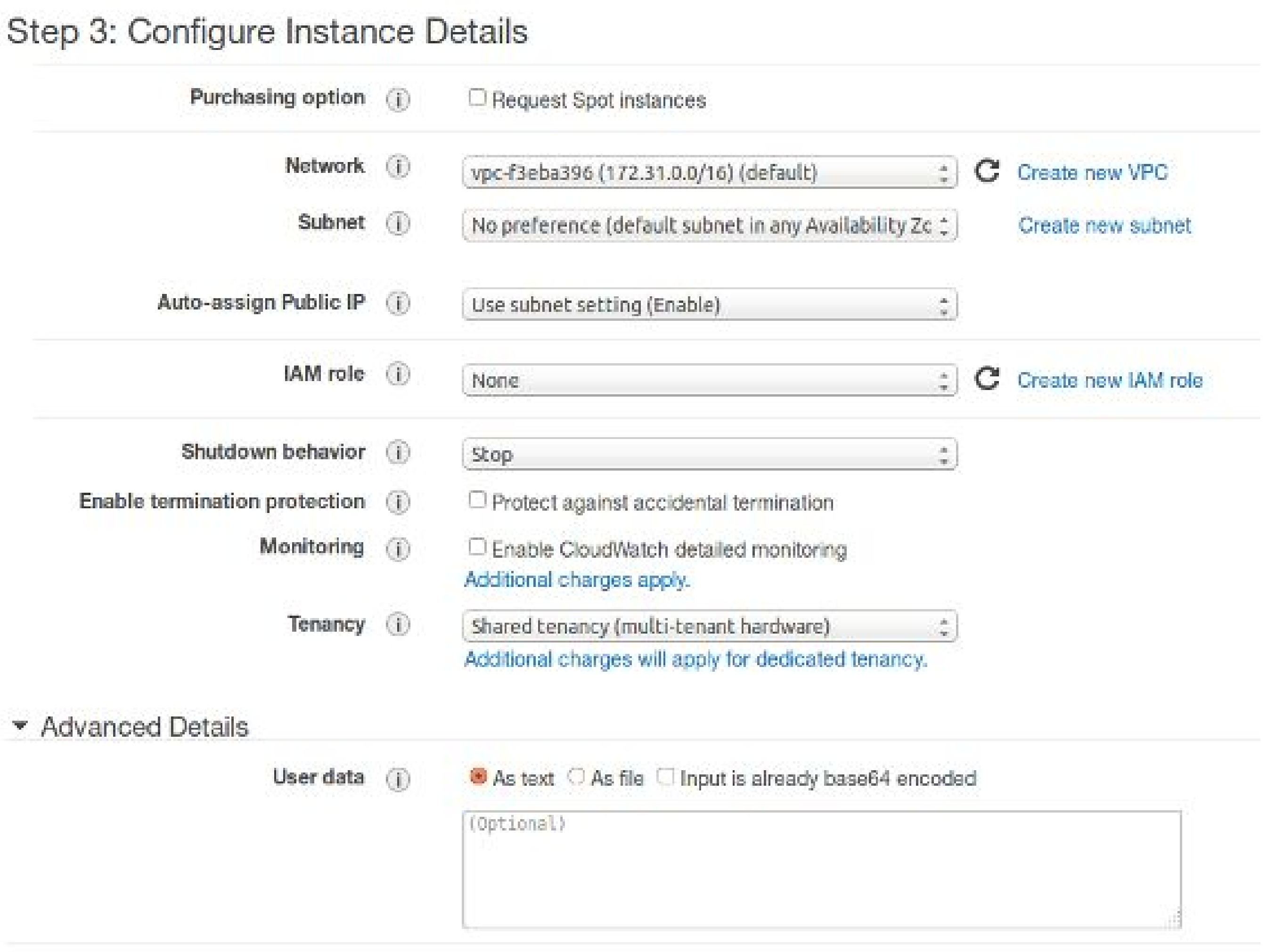
OS Patch Management

Applying the latest security patches to your instances is, again, your responsibility – even if you built your instance from an AWS defined AMI. No matter which operating system you deploy, I recommend that you regularly download the latest security patches. New bugs and security flaws are being discovered and fixed all the time, and you just can’t afford to ignore them.

I also suggest that you apply the latest patches immediately after creating an instance. You could look at automating this process through instance User data when creating your instances. For example entering the following on a Linux based AMI would automatically perform a Yum update at instance launch:

```
#!/bin/bash
yum update
```

The ‘User data’ section can be found in ‘Step 3: Configure Instance Details’ under Advanced Details:



Step 3: Configure Instance Details

Purchasing option	☐ Request Spot instances
Network	vpc-f3eba396 (172.31.0.0/16) (default) — Create new VPC
Subnet	No preference (default subnet in any Availability Zc — Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
IAM role	None — Create new IAM role
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring. Additional charges apply.
Tenancy	Shared tenancy (multi-tenant hardware). Additional charges will apply for dedicated tenancy.

Advanced Details

User data	[● As text] ○ As file ☐ Input is already base64 encoded
	(Optional)

Ensuring the latest patches are installed on your instances, protects you from vulnerabilities and threats to your OS. This is a simple yet necessary security addition to your instance rollouts.

Shared Tenancy vs Dedicated Tenancy

When deploying your Instances through the Console you will be asked which ‘Tenancy’ type you want for your instance:

Your two choices are ‘Shared Tenancy (Multi-Tenant Hardware)’ or ‘Dedicated Tenancy (Single-Tenant Hardware)’.

Shared Tenancy is when your instance will be hosted on shared hardware. This means there might be other AWS customers running their instances on the same physical server. You will never know who those customers are or how many of them there might be, but they will be equally ignorant about you. Security and separation are managed at the Hypervisor layer, where AWS maintains operational control and support. AWS guarantees that there will be no data crossover between account resources.

The major advantage of Shared Tenancy is its lower cost. This is the product of the fact that there is no need for AWS to isolate hardware for your explicit use…which, incidentally, is the definition of '**Dedicated Tenancy**'. If you are not happy sharing hosts, or need additional physical security separation, then select Dedicated Tenancy.

Not all instance types are eligible for Dedicated Tenancy, so if you are thinking of using it, consult the AWS documentation.

EC2 Key Pairs

If you plan to remotely log into your EC2 instances, then you'll need to create key pairs and associate them with your instances. A key pair consists of a public key and a private key. The public key is kept on your instance, while the private key must be available only to you, and will generally live on your local PC. AWS will not keep a copy of the private key. You will only be able to directly connect to your instance by invoking the private key. Public-key cryptography is used to encrypt/decrypt both keys.

These keys provide an added layer of security ensuring only people and resources holding the private key are allowed to make API calls to the instance. I suggest you download and keep a secure copy of your private key when prompted during your instance launch, as you will not be allowed to access your instance if you lose it.

When creating a new EC2 instance, you must specify which key pair you wish to associate. If you want to use a brand new key pair, you can create and configure one during this selection process.

There are a number of ways to create new key pairs. You can use the AWS Console, the AWS CLI, or Windows PowerShell. AWS uses 2048-bit SSH-2 RSA keys, with a limit of up to 5000 pairs per AWS Region.

Should you wish to create your own key pairs outside of AWS, they must be RSA compliant. EC2 will accept OpenSSH, Base64 encoded DER, and SSH Public key formats as per RFC4716. More information on how to create and import your own Keys can be found here.

Network Level Security

VPC Subnets

Segmenting your VPC into different networks is important both from a deployment/design perspective and also for security. Segmentation allows you to better refine your security profile as appropriate for each of the services operating within each subnet.

A subnet is a distinct network segment with its own IP address range within the larger VPC CIDR (Classless Inter-Domain Routing) range. As an example, if your VPC was created with a CIDR range of 172.16.0.0/16, you could choose to create subnets with a /24 mask allowing you to create networks – each with 256 available IP addresses – following this pattern: 172.16.1.0/16, 172.16.2.0/16, 172.16.3.0/16...

Note, that the first four addresses (between .0 and .3) and the last IP address (.255) of any subnet are not available, as they are reserved for router, DNS, Broadcast, and Network addresses, along with future capabilities. This leaves you with 252 addresses to use in the above example in each subnet. You can always use one of the many subnet calculators available online to help you plan and manage your addressing.

When planning your subnets, take the time to properly anticipate how many nodes you might actually need in the future: once you create a subnet, it can't be resized. Also, remember that AWS won't let you create masks lower than /16 or higher than /28.

Public and Private Subnets

Splitting up your CIDR address space into subnets means that each subnet will have its own ACL and Routing Table. You will have to decide whether each subnet should be public or private. A public subnet is defined by its connection (through a Routing Table) to an Internet

Gateway (IGW) within your VPC . The IGW provides a doorway from your environment out to the Public Internet. Any subnet without a route to the IGW is considered private, as there is no practical way for instances on this subnet to directly reach the public Internet.

When setting up your infrastructure, you will most likely be deploying services across multiple regions and availability zones (AZs) for high availability. It's important to remember that subnets are unique and can't be deployed across multiple AZs. If you plan to implement VPC peering, you should be aware that the peered subnets must not use overlapping CIDR blocks!

For tighter security, I would suggest keeping the number of instances within your public subnet(s) to a minimum, using Elastic Load Balancing and Autoscaling for increased availability while, at the same time, minimizing exposure. Only place instances that require public access in your public subnet. ALL other instances – like back end databases or application services – should live within private subnets. You will, of course, need to ensure proper routing between the two subnets.

The AWS Network ACL

AWS Network ACLs (NACL) are the network equivalent of the security groups we've seen attached to EC2 instances. NACLs provide a rule-based tool for controlling network traffic ingress and egress at the protocol and subnet level. In other words, ACLs monitor and filter traffic moving in and out of a network.

You can attach an ACL to one or more subnets within your Virtual Private Cloud (VPC). If you haven't created a custom ACL, then your subnets will automatically be associated with your VPC's default ACL which 'Allows' all traffic to flow into and out of the network.

You may notice that the AWS Network ACL rule base works much the same way as the rules within security groups. However, ACL rules include an additional field called 'Rule #', which allows you to number your rules. This is important, because ACL rules are read in ascending order, with each rule applied against matching packets regardless of whether a later rule might also match. For this reason, you will want to carefully sequence your rules with an organized numbering system.

You can number your rules starting at one '1' (which will be read first), and going as high as 32766. I would suggest that you leave a gap of at least 50 numbers between each of your rules (i.e., 50, 100, 150…) to allow you to more easily add new rules in sequence later, if it should be necessary.

Each list includes a final entry with an asterisk (*) in the 'Rule #' column, rather than a number. This rule appears at the end of every rule base and can not be removed or edited. Its job is to act as an automatic fail-safe, to ensure that traffic that doesn't match any of your custom ACL rules is dropped.

You will remember from the section above that security groups are stateful by design. NACLs, on the other hand, are stateless. Therefore, when creating your rules, you may need to apply an outbound reply rule to permit responses to inbound requests – if desired.

Let's explain these fields, one at a time.

Rule #. As we mentioned, ACL rules are read in ascending order, but only until a rule matching the packet is found. Care should be taken to number your rules appropriately when creating your rule base.

Type. This drop down list allows you to select from a list of common protocol types, including SSH, RDP, HTTP, and POP3. You can alternatively specify custom protocols such as varieties of ICMP.

Protocol. Based on your choice for ‘Type’, the Protocol option might be greyed out. For custom rules like TCP/UDP however, you should provide a value.

Port Range. If you do create a custom rule, you’ll need to specify the port range for the protocol to use.

Source. This can be a network subnet range, a specific IP address, or even left open to traffic from anywhere (using this value: 0.0.0.0/0).

Allow/Deny. Each rule must include an action specifying whether the defined traffic will be allowed to pass or not.

Network ACL Limitations

When creating your ACLs be aware that there is a default limit of 20 inbound and 20 outbound rules per list. You can request a higher limit from AWS, but the absolute maximum is 40, and network performance could be affected by any increase. There is also a top end limit of 200 ACLs per VPC.

All AWS Network ACL configurations – including adding and deleting rules and subnet associations – can also be applied using the AWS CLI, PowerShell, and AWS EC2 CLI.

As you can see, there’s nothing complicated about implementing ACLs, so I would definitely recommend you take a look at your own setup to see if you can improve it. You should at least try to avoid the default ACL setting that allows all traffic through using all protocols: this is very insecure for a live production environment.

I have seen NACLs used very effectively to prevent DDOS attacks. If, for example, traffic somehow manages to get past AWS’s own DDOS protection undetected and you are now being attacked from a single IP address – say 186.123.12.24 – you can create an NACL rule that will drop all traffic from that source. For better performance, I would place this rule at very start of your rule base.

Your NACLs will require updating from time to time, and you should regularly review them to ensure they are still optimized for your environment. Network security is an ongoing struggle and needs our regular attention to ensure its effectiveness.

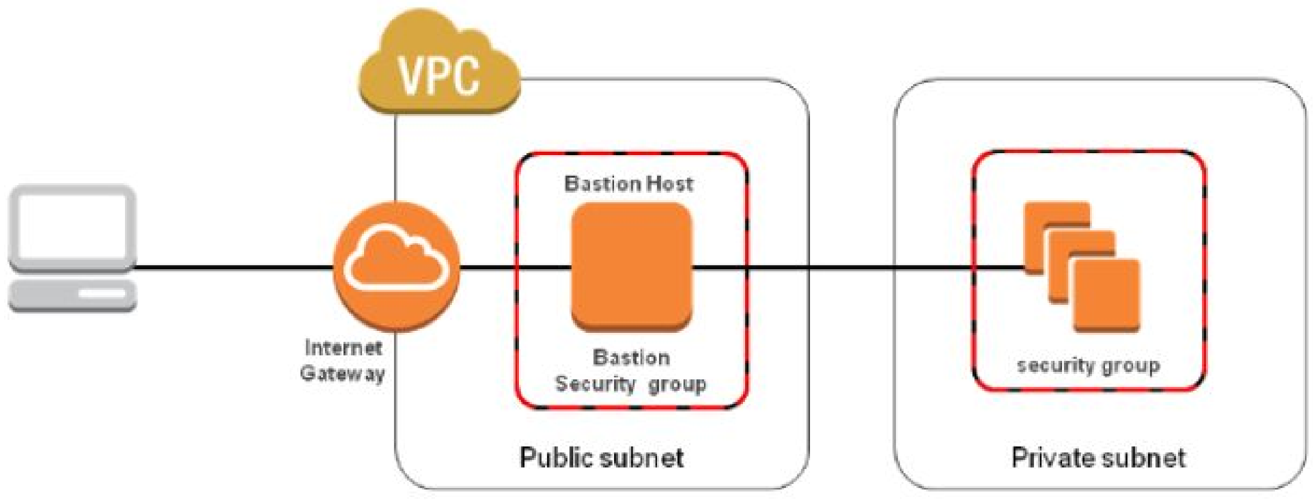
You can take the Network level security that much further by implementing more advanced configurations, such as Bastion Hosts, NAT instances and VPC Peering.

Bastion Hosts

Bastion hosts are instances that sit within your public subnet and are typically accessed using SSH or RDP. Once remote connectivity has been established with the bastion host, it then acts as a 'jump' server, allowing you to use SSH or RDP to login to other instances (within private subnets) deeper within your network. When properly configured through the use of security groups and Network ACLs, the bastion essentially acts as a bridge to your private instances via the Internet.

You may ask yourself, "Do I need one of those in my environment?" If you require remote connectivity with your private instances over the public Internet, then I would say yes!

This diagram shows connectivity flowing from an end user to resources on a private subnet through an bastion host:



When designing the bastion host for your AWS infrastructure, you shouldn't use it for any other purpose, as that could open unnecessary security holes. You need to keep it locked down as much as possible. I would suggest you look into hardening your chosen operating system for even tighter security.

Here are the basic steps for creating a bastion host for your AWS infrastructure:

- Launch an EC2 instance as you normally would for any other instance.
- Apply your OS hardening as required.
- Set up the appropriate security groups (SG).
- Implement either SSH-Agent Forwarding (Linux connectivity) or Remote Desktop Gateway (Windows connectivity).
- Deploy an AWS bastion host in each of the Availability Zones you're using

Security groups are essential for maintaining tight security and play a big part in making this solution work. First, create a SG that will be used to allow bastion connectivity for your existing private instances. This SG should only accept SSH or RDP inbound requests from your bastion hosts across your Availability Zones. Apply this group to all your private instances that require connectivity.

Next, create a security group to be applied to your bastion host. Inbound and outbound traffic must be restricted at the protocol level as much as possible. The inbound rule base should accept SSH or RDP connections only from the specific IP addresses (usually those of your administrators' work computers). You definitely want to avoid allowing universal access (0.0.0.0/0). Your outbound connection should again be restricted to SSH or RDP access to the private instances of your AWS infrastructure. An easy way to do this is to populate the 'Destination' field with the ID of the security group you're using for your private instances.

SSH and RDP connections require private and public key access to authenticate. This does not pose a problem when you are trying to connect to your bastion host from a local machine, as you can easily store the private key locally. However, once you have connected to your bastion host, logging in to your private instances from the bastion would require having their private keys on the bastion. As you will probably already know (and if not, then take careful note now), storing private keys on remote instances is not a good security practice.

As a result, I suggest that you implement either Remote Desktop Gateway (for connecting to Windows instances) or SSH-agent forwarding (for Linux instances). Both of these solutions eliminate the need for storing private keys on the bastion host. AWS provides great documentation on how to implement Windows Remote Desktop Gateway and SSH-agent forwarding.

As with all cloud deployments, you should always consider the resiliency and high availability of your services. With this in mind, I recommend deploying a bastion within each Availability Zone that you are using. Remember: if the AZ hosting your only AWS bastion host goes down, you will lose connectivity to your private instances in other AZs.

NAT instances for AWS infrastructures

A NAT (Network Address Translation) instance is, like an bastion host, an EC2 instance that lives in your public subnet. A NAT instance, however, allows your private instances outgoing connectivity to the Internet, while at the same time blocking inbound traffic from the Internet. Many people configure their NAT instances to allow private instances to access the Internet for important operating system updates. As I've discussed previously, patching your OS is an important part of maintaining instance level security.

AWS provides two ways of implementing a NAT host within your VPC, either through a NAT gateway, or by configuring a NAT instance. Using a NAT gateway offers you a managed implementation that offers increased bandwidth and availability, whilst at the same time requiring less configuration and operation from you. More information on NAT Gateway can be found here.

If you choose to manage and configure a NAT instance inside your VPC yourself, the following steps have to be taken:

- Create a Security Group which will be applied to your NAT.
- Select a predefined AMI and configure it as you normally would any other EC2 instance.
- Set up correct routing.

- Once your NAT is launched, it's important to disable source/destination checks. To do this, right click on your NAT Instance within the AWS Console and select 'Networking > Change Source/Dest. Check > Yes, Disable'.
- When creating a security group for your NAT, make sure that you allow inbound traffic from your private instances through the HTTP (80) and HTTPS (443) ports to allow for OS and software updates. Your outbound rule set should have an open destination of 0.0.0.0/0 for port 80 and 443 as well. If your instances will require any other ports opened, this is where to do it.

AWS provides some Amazon Machine Images (AMIs) that are already pre-configured as NAT instances – I recommend you consider using one. NAT AMIs have names that include the string 'amzn-ami-vpc-nat' so they're easy to find by searching from the Community AMI tab in Step One when launching an EC2 instance. These AMIs are a good idea, as they're configured right out of the box for IPv4 forwarding and iptables IP masquerading. ICMP redirects are disabled.

You will now have to modify the route table used by your private subnets. Make sure you have a route pointing to the outside world (0.0.0.0/0) via your new NAT instance. Your NAT-instance must be launched within your public subnet and have a public IP address. The route table of your public subnet where your NAT resides must have a route to the outside world by way of your Internet Gateway. This will ensure that any request from your private instance will first go to the NAT, and the NAT will forward that traffic out via the IGW to the Internet.

Your NAT is now set up and your private instances should be able to communicate with the outside world for updates etc., using ports 80 and 443. However, it's important to note that connections *initiated from the Internet* will not reach your private instances – as this configuration protects them.

Introduction to AWS VPC peering (Virtual Private Cloud)

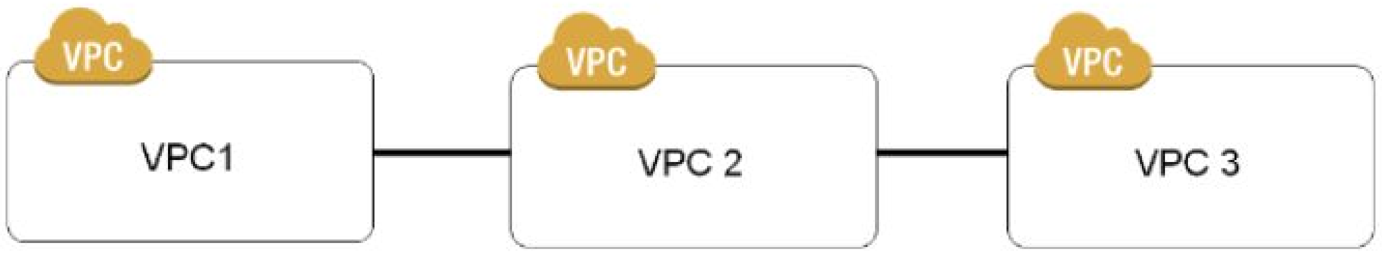
AWS VPC Peering lets you connect two VPCs together as a single network. It allows you to share resources between AWS VPCs without routing data through the Internet or a VPN connection. AWS VPC peering provides a tight and secure shared environment while minimizing external exposure.

Peered VPCs communicate across their private IP blocks, therefore it's important to ensure that the two VPCs do not have overlapping CIDR Address ranges. It's also important to note that you are unable to directly reference a security group from one VPC to the other. Instead, you'll need to enter a CIDR Block or specific IP address in the Source/Destination section of your SG rules.

AWS VPC peering design

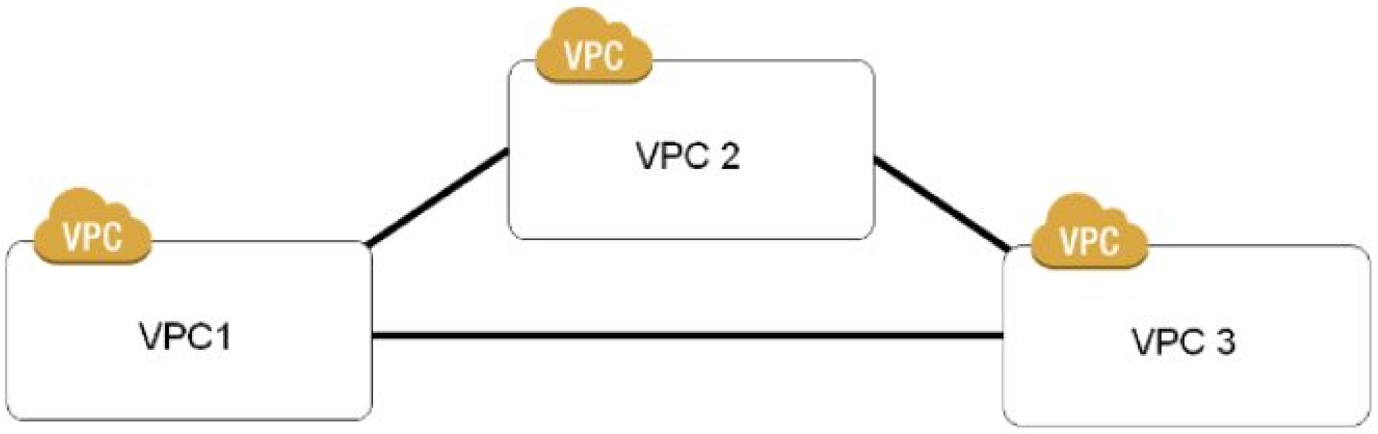
When setting up a peered connection, one VPC acts as the requester (the VPC initiating the connection) while the other acts as a peer. Before a connection can be established, the owner of the peer VPC has to acknowledge the request and accept the Peering connection. Once a connection has been established, routing between the CIDR blocks of each VPC needs to be added to a route table to enable resources within the networks to talk to each other via the private IP address range.

From a design perspective, you are not able to daisy chain VPCs together expecting them all to talk together across one large network. Each AWS VPC will only communicate with its 'requester' or 'peer'. As an example, if you have a peering connection between VPC 1 and VPC 2, and another connection between VPC 2 and VPC 3 as below:



...Then VPC 1 and VPC 2 could communicate with each other directly, as can VPC 2 and VPC 3. However, VPC 1 and VPC 3 could not. You can't route through one VPC to get to another.

To allow VPC 1 and VPC 3 to talk directly, you would have to implement a separate peering connection between VPC 1 and VPC 3 as shown below:



AWS VPC peering provides an excellent secure and trusted connection between VPCs for enhanced management and resource sharing. Depending on the way you've configured your VPCs, there may be many reasons why you would want to incorporate such an architecture into your environment. AWS offers some possible scenarios of their own that are definitely worth exploring.

Identity & Access Management (IAM)

AWS Identity and Access Management (IAM) combines with multi-factor authentication for a powerful and secure solution. Let's explore using Roles, Groups, and Users for AWS identity and access management.

Identity and Access Management (IAM) is one of the key security services within AWS. This service primarily governs and controls user access to your AWS resources. It achieves this through Users/Group/Roles and Policies. IAM uses a number of critical access methods such as MFA (Multi-factor Authentication) to manage federated identities.

I believe that IAM is an impressively powerful service that must be understood.

There is a distinction between Users, Groups, and Roles. Understanding the difference is pivotal in your deployment of access security within your environment. All of these elements are created and configured through IAM.

IAM Users are account objects that allow an individual user to access your AWS environment with a set of credentials. You can issue user accounts to anyone you want to view or administer objects and resources within your AWS environment. Permissions can be applied individually to a user, but the best practice for permission assignments is to assign them via the use of Groups.

IAM Groups are objects that have permissions assigned to them via Policies allowing the members of the Group access to specific resources. Having Users assigned to these groups allows for a uniform approach to access management and control.

IAM Roles are again objects created within IAM which have Policy permissions associated to them. However, instead of being associated with Users as Groups are, Roles are assigned to instances at the time of launch. This allows the instance to adopt the permissions given by the Role without the need to have Access Keys stored locally on the instance, which as we now know is bad practise.

IAM Users

When you first create your AWS root account, that account will have Administrative privileges and you will be able to access everything within your environment. **I strongly recommend you do *not* use this account to perform day to day tasks within your environment.** Instead, I encourage you to create a new user account with the required privileges to perform your work. If you need to create further Administrator users, keep it to a minimum for obvious reasons and place them in a single Group.

When creating your User accounts, or any object for that matter, ensure you have a sensible naming convention and adhere to it. This will help with identification of objects going forward throughout IAM and other services. Only create users who genuinely require access to your AWS infrastructure, those who have a purpose such as providing operational support for your project, for example.

Before you come up with a naming convention for your accounts be mindful that that you are not able to use the following characters + = , . @_-. There is also a limit to the number of users who can be associated within your AWS account — which is currently set to 500 Users.

You can allow AWS to generate its own password for the user or you can choose to manually create your own. Either way, you can optionally allow users to change their passwords on their first login. Depending on current security policies, you may choose to select this option or not.

If you allow AWS to automatically assign the password, the system will use the options specified in your AWS Password Policy. This policy can me modified to suit your own security requirements, adapting it to fit your existing needs.

IAM Groups

Groups within IAM are objects that allow you to efficiently manage permissions and access your resources within your AWS environment. Using Groups to control permissions is the desired best practice from a management perspective. This is especially important if you have a large number of users to administer/control. Another advantage of this best practice is when a user changes roles or departments and their responsibilities change, it's easy to remove them from one group and add them to another without worrying about individual access policies.

Group permissions are governed by policies. There are excellent predefined policies supplied by AWS, and you can always create your own if the predefined policies don't suit your needs. The policies are scripts that dictate what resources can be accessed and by whom. The policy/script is then attached to a group whereupon users are made members of that group and adopt those set permissions.

When associating Users with a Group, there is an upper limit of 10 groups per User. You are also limited to 100 groups per AWS account — so be sure to plan your access carefully.

IAM Roles

Roles are similar to Groups in that they are objects created within IAM. Instead of providing permissions for Users, they provide permissions for instances, such as EC2. Policies are associated with Roles and during instance creation a Role can be selected to grant that instance access to what the associated policy dictates.

The great benefit of assigning roles to Instances is that it avoids the need to store Access Keys for API calls from the local instance. Storing any kind of access key locally on an instance is a **bad practice** (especially if you have key information hardcoded within any code held locally on the instance). As a result, Roles allow instances to adopt the permissions required without the use of the access keys being assigned.

As with Users and Groups, there are some associated limitations to Roles. Note that an instance can only be associated with **ONE** role, but you have the capacity for up to 250 Roles within your AWS Account.

When an instance is launched with a Role attached, the EC2 instance will make any API request to the services it has access to without the need of Access Keys. This is a great way to apply permissions to resources.

Take note that you are not able to change the Role of an Instance once it is launched. If you need to change the Role, you must create a new instance and select the new Role. *Depending on your circumstances, it may be a good idea to create an AMI of your existing Instance first and then create a new instance using this AMI before selecting the new Role.*

Multi-factor Authentication

Multi-factor Authentication (MFA) adds an additional layer of security through the requirement of additional credentials via an MFA device (typically a 6 digit number) on top of the username and password. This number is a randomly generated single-use code that lasts for a very short period of time. At minimum, MFA should be employed for users who have the greatest privileges, such as Administrators and Power Users.

AWS does not charge for the use of MFA on activated User accounts, but you will need a 3rd party MFA device whether it's a virtual or physical one. AWS provides a summary of all supported devices here. Personally I use Google Authenticator for the iPhone because it is simple and easy to setup & configure.

IAM Policies

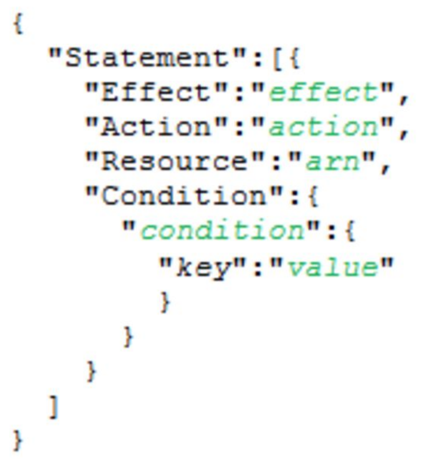
Using a predefined IAM policy is more likely than not a perfect match for the permissions you actually need. However, now and again, you may want to tweak a small part of it to more exactly fit your requirements.

Remember, when it comes to security, the more precisely you can define your controls, the better. You should avoid being lazy and deploying implementations that are "close enough" rather than hand crafting the perfect fit.

An IAM Policy is a JSON script made up of statements following a set syntax for allowing or denying permissions to an object within your AWS environment.

IAM Policy Syntax

Each policy has to have at least one statement whose structure might look like this:



```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

The **‘Effect’** element can be set to either ‘Allow’ or ‘Deny’. These are explicit. By default, access to your resources is denied and so therefore if this is set to ‘Allow’ it replaces the default ‘Deny’. Similarly, if this were set to ‘Deny’ it would override any previous ‘Allow’.

The **‘Action’** Element corresponds to API calls to AWS Services that authenticate through IAM. For example, this example represents API calls to delete a bucket (the action) within S3 (the service):

“Action”:”s3:DeleteBucket”

You are able to list multiple actions, if required, by using a comma to separate them:

“Action”:”s3:DeleteBucket”,”s3:CreateBucket”

Wildcards (*) are also allowed. So, for example, you could create an action to carry out all APIs relating to S3, such as:

“Action”:”s3:*”

The **‘Resource’** element specifies the actual resource you wish the permission to be applied to. AWS uses unique identifiers known as ARNs (Amazon Resource Name) to specify resources. Typically ARNs follow a syntax of:

Arn:partition:service:region:account-id:resource

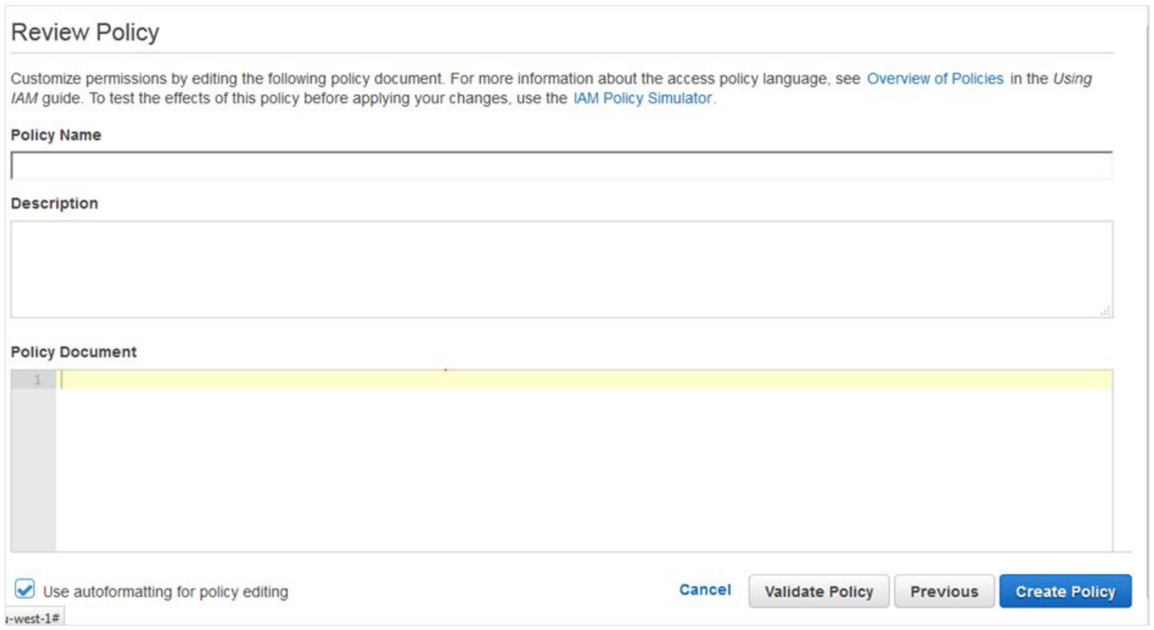
- ‘Partition’ – This relates to the partition that the resource is found in. For standard AWS regions, this section would be ‘aws’.
- ‘Service’ – This reflects the specific AWS service, for example ‘s3’ or ‘ec2’.

- ‘Region’ – This is the region where the resource is located. Some services do not need a region specified, so this can sometimes be left blank.
- ‘Account-ID’ – This is your AWS Account ID (without hyphens). Again, some services do not need this information, and so can be left blank.
- ‘Resource’ – The value of this field will depend on the AWS service you are using. For example, if I were using the Action: “Action”:”s3:DeleteBucket”, then I could use the bucket name that I wanted the permission to delete: arn:aws:s3:::cloudacademyblog.

The **‘Condition’** element of the IAM Policy is an optional element that allows you to specify when the permissions will be activated based upon set conditions. Conditions use key value pairs, and all conditions must be met for the permissions to be activated. A full listing of these conditions can be found here.

Writing your own IAM policy

If you’re ready to write your own IAM policy from scratch, there’s nothing stopping you. From the AWS Console, select ‘**IAM > Policies > Create Policy**’, and this time select ‘**Create your own policy**’.


Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see Overview of Policies in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the IAM Policy Simulator.

Policy Name

Description

Policy Document

☑ Use autoformatting for policy editing

Here you are presented with a blank canvas allowing you to give your policy a name and description…but most importantly, a blank policy document where you can write your JSON document adhering to the policy syntax rules discussed at the beginning of this article.
Once you are happy with your policy, use the 'Validate Policy' button at the bottom to determine if there are any errors with your policy. Again, when you are finished click '**Create Policy**'.

To save you writing a complete policy from scratch, you are able to copy an existing policy and then edit the content of the JSON script with the small changes that you require. This can save you considerable time if you are aware of an existing policy that closely resembles your requirements.

In addition to this, if you do not feel confident in writing your own policies you can use the Policy Generator from within the IAM Service to build your policies through a series of drop down boxes.

Once you have written your Policies you can you use the built in Policy Simulator to test your policies and ensure they behave as you expected.

As you can see there are a number of methods available to help you refine your IAM policies, and you aren't forced to use predefined policies. Once you get used to the syntax – and benefits – of writing your own policies, you will be able to effectively and efficiently lock down access to your resources to ensure they are only accessed by authorized API calls.

There are many, many specific commands that can be controlled through an **IAM policy**, but they're a bit beyond the scope of this eBook. AWS provides great API listings for the different services through their extensive documentation for advanced policy writers.

Federated Access

As we know we can create users and groups within IAM that can be assigned permissions to access specific resources. What if you don't want to have to create new IAM accounts for potentially hundreds or more of your personnel? Wouldn't it be better to use your company's existing Active Directory accounts instead? Or another example might be, perhaps you are developing a mobile app/game and using AWS to host your infrastructure and you want/need the users to authenticate with their Facebook or Google accounts. What is the best way of managing this process — which could involve potentially millions of end users? Federated AWS access is the solution to these scenarios.

What is Federated Access?

Federated access simply allows external entities to temporarily connect and access AWS resources without requiring an existing IAM user account. Unlike the restriction with IAM users, there are no limits on the number of federated users you can have.

Typically AWS supports two methods of Federation, **Enterprise Identity Federation** and Web Identity Federation. Enterprise Identity Federation allows your existing Active Directory users to authenticate to your AWS resources, allowing for a Single sign-on (SSO) approach. Web Identity Federation allows you to make mobile apps created on AWS that integrate with Public Identity Providers (IdPs) for authentication such as Facebook, Google or any OpenID Connect provider.

There are different ways to configure your federated users depending on your requirements. If you search, you will find a host of blog entries showing how to do this — many of the posts from AWS.

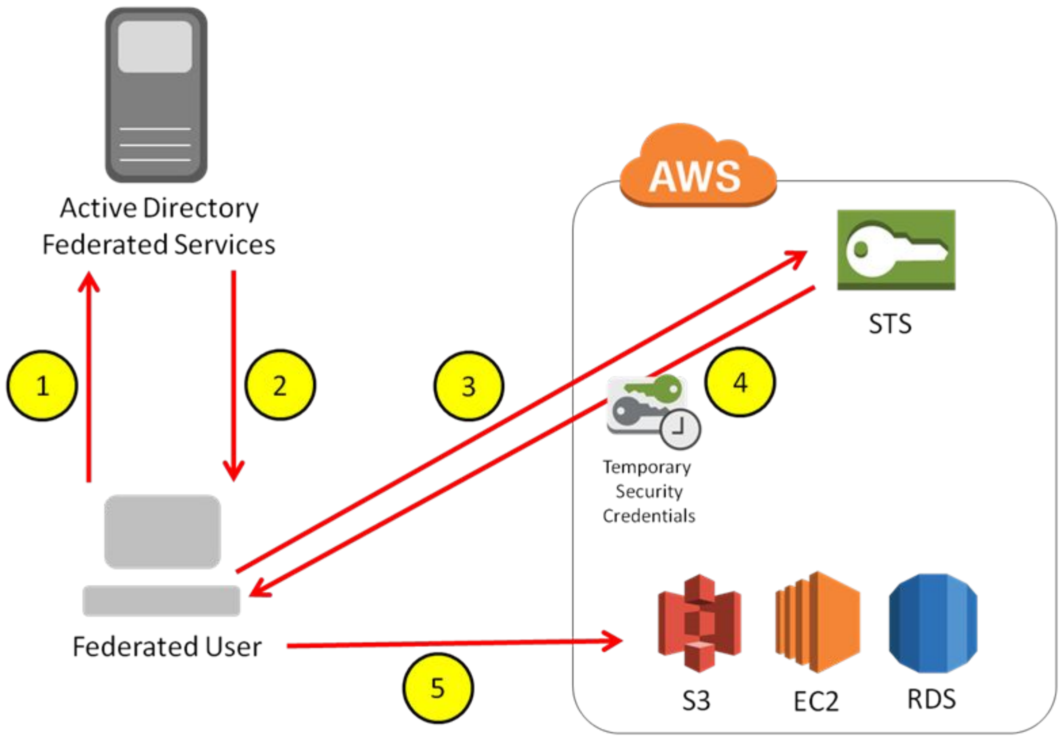
I will give an overview of how federation within an Active Directory situation should look, and how it works. I'll show the high-level steps involved in how AD allows this temporary access in conjunction with AWS. For this example, federation within Active Directory has integration with SAML 2.0 (Security Assertion Markup Language) and with the Security Token Service (STS) from AWS.

SAML 2.0 allows for SSO through the exchange of access credentials to be passed between an identity provider (IP) and a service provider (SP) through the means of an Assertion.

The Security Token Service (STS) is an AWS web service that allows you to gain temporary security credentials for federated users via IAM benefiting from existing access roles and policies. More information on STS can be found here on an Amazon Web Services page.

Active Directory Federated Access overview

The diagram below shows the typical steps involved in authenticating a federated user to access AWS services.



1. The user initiates an authentication against the Active Directory Federated Service (ADFS) Server via a web browser to the SSO URL

2. If authentication is successful via Active Directory credentials, SAML (Security Assertion Markup Language) will then issue an assertion back to the users client

3. The SAML assertion is then sent to the Security Token Service (STS) within AWS to assume a role within IAM using the AssumeRoleWithSAML API (more on this API can be found on this AWS Page)

4. STS then responds to the user requesting federated access with temporary security credentials with an assumed role and associated permissions

5. The user is then has federated access to the necessary AWS services as per the Role permissions

This is a simple overview of how the federation is instigated from the user. Corporate Identity Federation is ALWAYS authenticated internally first by Active Directory before AWS.

There are a number of configuration steps required to get to the diagram above implemented and for a complete guide and lengthy step by step process about setting up Enterprise Identity Federation (including the installation of the ADFS Server etc, please click here. It discusses how to set up your Active Directory Groups that map to your pre-defined IAM Roles (Step 3 above) for users that do not have an IAM account, resulting in a Single Sign-on method to your AWS infrastructure.

If you are looking to configure Web Identity Federated Access please click here. This will provide you very specific instructions on how to configure this setup.

AWS Trusted Advisor

The AWS Trusted Advisor is like having your own AWS specialist constantly looking over your shoulder ensuring you are keeping an eye on your infrastructure for optimal performance, security and best practises. It provides prompts and suggestions that you should make to your AWS environment for added security, such as ensuring your root account is using Multi-Factor Authentication (MFA). It can also allow you to save costs by way of notification that certain services/instances haven't been used for some time and could potentially be powered down.

Although every AWS accounts is given some free checks within Trusted Advisor, the full power and potential of the service can only be appreciated with the full access to all 47 checks. If you have a large estate to manage, monitor and assess for best practises then I suggest you take full advantage of the Trusted Advisor service within the corresponding support plan. It could help you save a significant amount of money by ensuring your security is optimised against best practises. It will check to ensure you have the right level of instances and that they are optimized for size. TA will also check your High Availability configurations to ensure they are in place in the event of a failure.

Typically in an ever expanding environment, it can be difficult to stay on top of all of these areas, so having a service to monitor your activities and spot things that you may miss is essential in keeping your estate optimised.

AWS Billing Controls

When it comes to monitoring and managing your AWS financial information, you need to ensure this privilege is restricted to the people who are authorised. Billing is linked with IAM to allow specific restrictions of access to be put in place.

By default, your AWS account owner has full access to all billing information. No other users initially have access. The first time that your root account tries to access the billing information you need to perform the steps below as it's not automatically activated and isn't initially visible.

1. Log into AWS Console as the AWS account owner (root)

2. Select your account name in the top right and select 'My Account'

3. Select 'Edit' next to IAM User Access to Billing Information and select the check box to activate

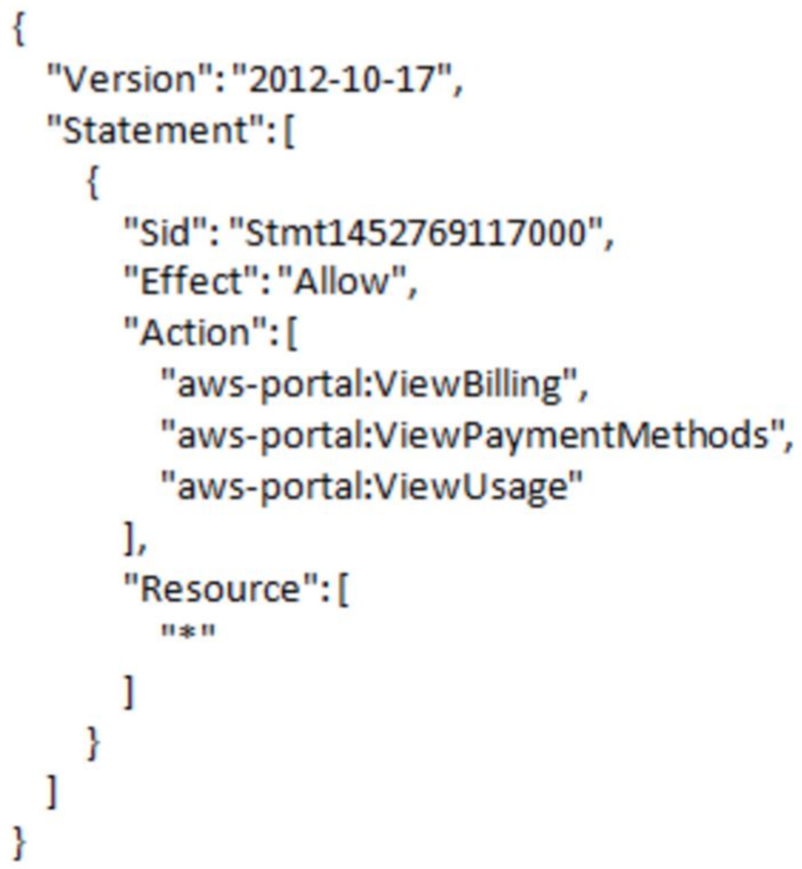
The root account can now access the Billing & Cost Management from within the console by selecting your username/account in the top right corner > Billing and Cost Management. From here you are able to access information such as your Bills, Budgets, Reports, Payment History and Method etc. Some very sensitive information.

If you require other users to access this restrictive area you have to grant specific permissions.

Allowing access to Billing information

You can allow permissions for users to access your billing information via IAM policies and permissions. You are able to refine access within Billing and Cost Management further by allowing some people to view specific information only while some users are allowed to modify certain aspects of the billing information. A full range of permissions and restrictions can be found here .

As an example I have created a policy below that allows viewing access to the ‘Billing’, ‘Payment Method’s and the ‘Usage section within the Billing and Cost Management Dashboard:



```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid": "Stmt1452769117000",
      "Effect":"Allow",
      "Action":[
        "aws-portal:ViewBilling",
        "aws-portal:ViewPaymentMethods",
        "aws-portal:ViewUsage"
      ],
      "Resource":[
        "*"
      ]
    }
  ]
}
```

This policy can then be applied to a group or specific user as required.

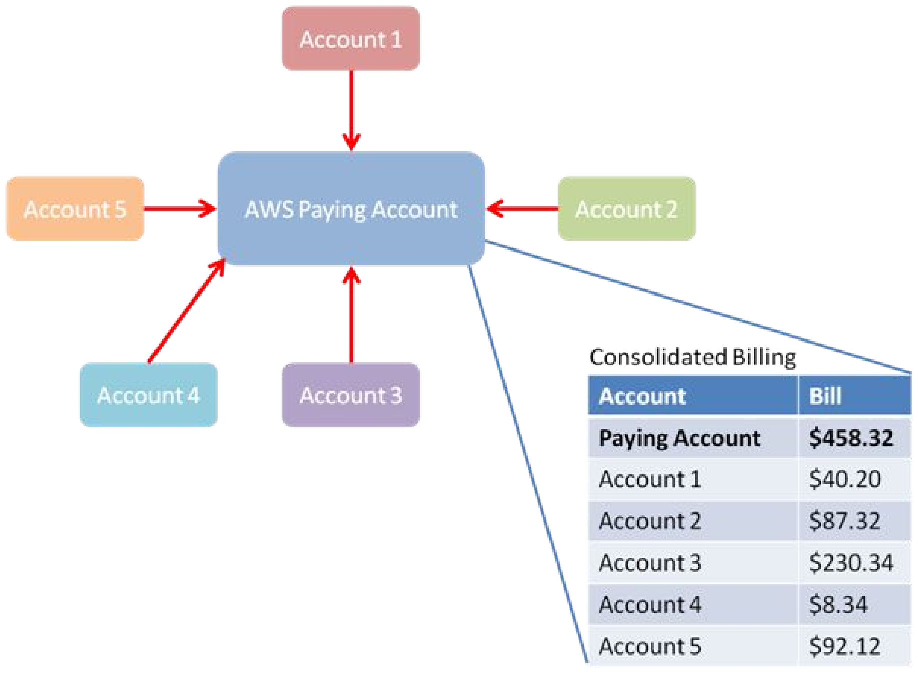
AWS Linked Accounts (Consolidated Billing)

Discussing AWS Linked Accounts follows on nicely from the previous topic of Billing. Linked Accounts allows you to securely link multiple AWS accounts to effectively consolidate all your AWS Billing via a single master Paying Account.

It's important to point out that by linking accounts, only the billing information is linked, no other information can be accessed between the accounts. Each account still acts completely independent of the other. One account is not able to view AWS services or data from another account.

If you are responsible for the finances across multiple accounts then you are probably aware how difficult this can be to manage. By consolidating the billing into one Paying Account you reap many advantages. It provides a simple overview of all your accounts each month. Only the Paying Account is charged, and from here you can see the usage of all other accounts in one place. You can even extract charges from separate accounts and export them to a .CSV file for further analysis.

The diagram below gives a graphical view of Consolidated Billing.



Consolidated Billing

Account	Bill
Paying Account	**$458.32**
Account 1	$40.20
Account 2	$87.32
Account 3	$230.34
Account 4	$8.34
Account 5	$92.12

If you link your account for consolidated billing, I strongly suggest that you enforce strong security on the Paying Account, and certainly implement MFA (Multi-Factor Authentication).

Simple Storage Service (S3) Security

If you are looking to implement security on S3 then you would already be familiar with what the service is and its reliability as a Storage service and the benefits it can bring. With this in mind you are probably storing a lot of data on this service and as a result you will want to ensure that it's safe and secure! I will run through some of the security elements of S3 that you can choose to deploy, depending on your data's sensitivity.

Bucket Policies

Bucket Policies are similar to IAM policies in that they allow access to resources via a JSON script. However, Bucket policies are applied to Buckets in S3, where as IAM policies are assigned to user/groups/roles and are used to govern access to any AWS resource through the IAM service.

When a bucket policy is applied the permissions assigned apply to all objects within the Bucket. The policy will specify which 'principals' are allowed to access which resources. The use of Principles within a Bucket policy differs from IAM policies, Principles within IAM policies are defined by who/what is associated to that policy via the user and group element. As Bucket policies are assigned to Buckets, there is this need of an additional requirement of 'Principles'. A Principal can be either another AWS Account, a Service or more commonly a User.

Setting Bucket Policy Conditions

Again similarly to IAM Policies, S3 Bucket Policies allow you to set conditions with the Policy, for example allowing specific IP subnets to access the Bucket and perhaps restricting a specific IP address.

For a full list of conditions and help on creating your S3 Bucket Policies take a look at the great tool that AWS provides here: AWS Policy Generator

An explicit deny within the policy will always take precedence over an 'allow'. Access of least privilege will always overrule where conflicts between policies exist. This will also be the case if you have an IAM user with S3 access to a specific bucket, which also happens to have a Bucket Policy. AWS will look at both policies and apply access on a least-privilege condition if there are conflicting permissions.

S3 Access Control Lists

In addition to IAM Policies and Bucket Policies, S3 also has an additional method of granting access to specific objects through the use of Access Control Lists (ACLs), allowing a more finely grained access approach than a Bucket Policy. ACLs allow you to set certain permissions on each individual object within a specific Bucket.

Again, access will always be granted on a least privileged condition if conflicts exist between ACLs, Bucket Policies and IAM Policies. ACLs can be managed and configured from within the S3 Service itself or via APIs.

An ACL on a Bucket/Object will have a Grantee, this is the resource owner and is likely to have Full Control over that object. This is typically the AWS Account owner.

Other permissions that can be set are List (Read), **Upload/Create** (Write), **View Permissions** and **Edit Permissions**. If all checkboxes are selected, that Grantee is considered to have Full Control of the object.

Additional Grantees can be added as per below:

- **Everyone** – This will allow access to this object by anyone, and that doesn't just mean any AWS users, but anyone with access to the Internet
- **Authenticated AWS Users** – This option will only allow IAM users or other AWS Accounts to access the object via a signed request
- **Log Delivery** – This allows logs to be written to the Bucket when it is being used to store server access logs
- **Me** – This relates to your current IAM AWS User Account

It is worth mentioning that an S3 ACL can have up to 100 Grantees.

There are slightly different permission options between a Bucket ACL and an Object. At the Object level you have the option to **Open/Download** the object, **View Permissions** and **Edit Permissions**.

Using S3 as an Origin for CloudFront (Content Delivery Network – CDN)

When you use S3 as your Origin for CloudFront everyone has Read permission for the objects in your bucket allowing anyone to access the content via the CDN. However, if anyone or an application has the unique URLs to the objects then this will bypass the features offered by CloudFront such as access times of that object and IP restrictions, this maybe a security concern that you want to alleviate.

To ensure that no-one can access your Origin Bucket unless they are going via your CDN then you can create an Origin Access Identity (OAI) and associate that with your CloudFront Distribution. You can then allow ONLY the OAI access to your Bucket using methods already discussed in this article and remove all other user access. This way, only users utilising CloudFront will be able to access the Origin Bucket and anyone with a direct URL link will be denied as only the OAI will have access.

Lifecycle Policies

Implementing Lifecycle policies within S3 is a great way of ensuring your data is managed safely (without experiencing unnecessary costs) and that your data is cleanly deleted once it is no longer required. Lifecycle policies allow you to automatically review objects within your S3 Buckets and have them moved to Glacier or have the objects deleted from S3. You may want to do this for security, legislative compliance, internal policy compliance, or general housekeeping.

Implementing good lifecycle policies will help you increase your data security. Good lifecycle policies can ensure that sensitive information is not retained for periods longer than necessary. These policies can easily archive data into AWS Glacier behind additional security features when needed. Glacier is often used as a 'cold storage' solution for information that needs to be retained but rarely accessed and offers a significantly cheaper storage service than S3.

Lifecycle policies are implemented at the Bucket level and you can have up to 1000 policies per Bucket. Different policies can be set up within the same Bucket affecting different objects through the use of object 'prefixes'. The policies are checked and run automatically — no manual start required. I have an important note on this automation, be aware that lifecycle policies may not immediately run once you have set them up because the policy may need to propagate across the AWS S3 Service. Very important when starting to verify your automation is live.

The policies can be set up and implemented either via the AWS Console or S3 API.

By implementing these policies it could ensure that I am not saving confidential (or sensitive) data unnecessarily. It would also allow me to reduce costs by automatically removing unnecessary data from S3 for me. This is a win-win.

I could choose to archive my data into AWS Glacier for archival purposes, I could have taken advantage of cheap storage ($0.01 per Gig) compared to S3. Doing this would allow me to maintain tight security through the use of IAM user policies and Glacier Vaults Access Policies which allow or deny access requests for different users. Glacier also allows for **WORM** compliance (Write Once Read Many) through the use of a Vault Lock Policy. This option essentially freezes data and prevents any future changes.

Amazon has good information on Vault Lock and provides more detailed info on Vault Policies for anyone who wants to dive a bit deeper.

Lifecycle policies help manage and automate the life of your objects within S3, preventing you from leaving data unnecessarily available. They make it possible to select cheaper storage options if your data needs to be retained, while at the same time, adopting additional security control from Glacier.

S3 Versioning

S3 Versioning, as the name implies, allows you to "version control" objects within your Bucket. This allows you to recover from unintended user changes and actions (including deletions) that might occurred through misuse or corruption. Enabling Versioning on the bucket will keep multiple copies of the object. Each time the object changes, a new version of that object is created and acts as the new current 'version'.

One thing to be wary of with Versioning is the additional storage costs applied in S3. Storing multiple copies of the same object will use additional space and increase your storage costs.

Once Versioning on a Bucket is enabled, it can't be disabled – only suspended.

Multi-Factor Authentication (MFA) Delete

As an additional layer of security for sensitive data, you can implement a Bucket Policy that requires you to have another layer of authentication to delete data, or change the version state of an object. This is activated via an authentication device which typically displays a random 6 digit number to be used in conjunction with your AWS Security Credentials. This additional layer of security is great if for some reason your AWS credentials are ever breached. If this happens, any hacker would still require your associated MFA Device to delete objects.

S3 Encryption

Data protection is a hot topic with the Cloud industry and any service that allows for encryption of data attracts attention. S3 allows you the ability of encrypting data both at rest, and in transit.

To encrypt data in transit, you can use Secure Sockets Layer (SSL) and Client Side Encryption (CSE). Protecting your data at rest should be done with Client Side Encryption (CSE) and Server Side Encryption (SSE).

Client Side Encryption

Client Side Encryption allows you to encrypt the data locally before it is sent to AWS S3 service. Likewise, decryption happens locally on the client side.

You have 2 options to implement CSE:
Option 1, use a Client Side master key.
Option 2, use an AWS KMS managed customer master key.

The Client Side master key works in conjunction with different SDKs, Java, .NET and Ruby. It works by supplying a master key that encrypts a random data key for each S3 object you upload. The data key encrypts the data, and the master key encrypts the **data key**. The client then uploads (PUT) the data and the encrypted data key to S3 with modified metadata and description information. When the object is downloaded (GET), the master key verifies which master to use to decrypt the object using the metadata and description information.
If you lose your master key, then you will not be able to decrypt your data. Don't let this happen to you.

AWS KMS Managed Customer Master Key

This method of CSE uses an addition AWS service called the Key Management Service (KMS) to manage the encryption keys instead of relying on the .SDK client to do so. You only need your CMK ID (Customer Master Key ID) and the rest of the encryption process is handled by the client. A request for encryption is sent to KMS, where the KMS service will issue 2 different versions of a data encryption key for your object. One version is plain text that encrypts your data and the second version is a cipher blob of the same key that is uploaded (PUT) with your object within the metadata.

When performing a GET request to download your object, the object is downloaded along with cipher blog encryption key from the metadata. This key is then sent back to KMS to return the same key but in plain text to allow the client to decrypt the object data.

Amazon provides detailed information on KMS (Key Management Service) so you can review before making and choices.

Server Side Encryption (SSE)

Server Side Encryption offers encryption for data objects at rest within S3 using 256-bit AES encryption (which is sometimes referred to as AES-256). One benefit of SSE is that AWS allows the whole encryption method to be managed by AWS if you choose. This option requires no setup by the end user other than specifying if you want the object to use SSE. You don't need to manage any master keys if you don't want to as you must with Client Side Encryption.

I say 'if you don't want to' as there are 3 options for SSE. One option is SSE-S3, which allows AWS to manage all the encryption for you including management of the keys. This can be activated within the AWS Console for each object.

Implementing AWS Security

I have only covered some of the AWS Security features and services within this short eBook and there are many more elements of AWS security, with more and more improvements released every month. This eBook can only act as a guide and you should always consult AWS documentation to ensure you are implementing the latest best practises from a security perspective.

If you have read this eBook and realised that you could benefit from some of these features discussed, then you have already realised you have a security threat and you should look to begin implementing these changes as soon as possible to prevent unauthorised access.

Although Security should be at the forefront of any new implementation and architect for at the beginning stages, many of these features can be implemented to existing AWS deployments, such as NACLs, Security Groups, IAM Policies, Bucket Policies, MFA, Bastion Host, NAT, etc.

Before implementing the different levels of security, ensure you plan them thoroughly to ensure you don't prevent any access this is actually required and as a result causing service interruption.

Add your layered security in a phased approach, do not implement multiple changes at once. Add one at a time and test, when you are happy with the desired results, then implement the next change. Adding multiple restrictions and changes at once can lead to troubleshooting being a huge headache, couple this with the stress of potential service down time and your stress levels will hit the roof. Adopt a phased installation, spending time to ensure its suitability and desirability.

How to Get Started

Cloud Academy has a huge range of courses, labs and blogs to help you begin understanding the best practices of implementing your security requirements. Use the links below to start securing your infrastructure:

Courses:
Data Security for Solutions Architect Associate on AWS
Security Fundamentals for AWS

Labs:
Securing your VPC using Public & Private Subnets
Introduction to Identity & Access Management (IAM)
Advanced Roles and Group Management using IAM
Learn the tools for AWS Governance

Learning Paths:
Fundamentals of AWS
Solutions Architect for AWS - Associate Certification for AWS
Solutions Architect for AWS - Professional Certification for AWS

Blogs:
Security Related Blogs

Security is the single and most fundamental aspect of a successful implementation and migration of any project within a Public Cloud environment. Often it is overlooked with the thought of 'I'll come back to it later' and this leads to vulnerabilities and mistakes within your infrastructure. Security should be at the forefront of any implementation and shortcuts surrounding security should never be taken.

Security needs to be inserted at every level of your solution, from your infrastructure stack through to your application stack. If you leave it out at any level, then you are exposing your services to an open risk which can lead to service failure, and in severe cases, loss of business and or customers.

Adding levels of security post installation can be done and should be done if it has been overlooked. This typically takes more time to plan and implement as any changes can affect your service levels, which is why it should always be considered at the planning stages. Chains are only as strong as their weakest links.

If you are an Architect, Project Manager, Engineer, or Technical Manager that is often in meetings defining the initiation of a project to the Cloud, be sure to bring up the topic of security at every stage and within every level. In a perfect world, it would be beneficial to have a Security Specialist on board to assist with planning and design.

Technology changes at near unimaginable rates. Growth is the goal and the challenge at times. Keeping up with constant changes is a difficult but essential challenge. You must be aware of new technologies and services that enable you to architect correctly while using best practises. The Cloud training and educational services offered by Cloud Academy will help you and your corporation stay at the forefront of these changes.

Cloud Academy is a great resource and offer a huge range of online **training videos** covering AWS, Azure, and Google Cloud. The courses provide theoretical overviews and clear demonstrations of tools and services you will use within the Cloud. In addition to this, there are a host of Labs allowing you to 'get your hands dirty' with a live console allowing you to deploy and configure services so that you can put your knowledge into practise. They offer a huge bank of quizzes to test your knowledge, reinforce learnings, and to see where you have skill gaps -- allowing you and your team to focus on key areas.

Focus differentiates Cloud Academy. They offer learning paths that take students step-by-step through the resources they need to certify in different disciplines. Their focus is evident in their approach to teaching and technology. Cloud Academy is all about the cloud.

CloudAcademy

JOIN IN!

cloudacademy.com