Understand AWS security groups

Let's take a look at how Amazon Web Services handles the important topic of network security using AWS Security Groups. First, we';; look at a traditional firewall setup. Imagine this is our data center. We have a few web servers, maybe a public API host. We also have some private hosts, things like databases that we want locked down and secure. In our simplified data center, we've divided our space into subnets: public on the left, private on the right. And because we're good system architects concerned about security, we've got a firewall around the whole thing. The firewall protects the whole data center at the border. We define the rules for what traffic goes in and out. We let the good traffic in, like our web users, which we allow in on ports 80 and 443. We lock down everything else, keeping the bad guys out. We also configure the firewall to let servers talk out to the world when appropriate, like when we need to call an external API. Of course, the firewall doesn't just protect the border. It can also govern traffic inside the data center, say between these subnets. These app servers need to talk to databases in the private subnet. So we write rules for that. If you're using an enterprise firewall writing point-to-point rules, this simple architecture could total up to one, two, three, four, five, six, seven, eight different rules. Over time, as the number of services and servers increases, this list of rules can become very long and hard to maintain. Even if we define broader rules, such as the public subnet is open to all world HTTP traffic, you still have a maintenance issue, because troubleshooting connectivity for a particular server means hunting down or knowing all the rules to find somewhere else that may affect it. In fact, sometimes it means you have to sit down and monitor the network traffic first, before you even know where to look for the relevant rules. AWS Security Groups are reusable groups of network security rules, and they define incoming and outgoing rules. They can be attached to almost any resource in AWS, such as EC instances, load balancers and RDS databases, and they act like local firewalls for that resource. AWS Security Groups deny all incoming traffic by default, and they allow all outgoing traffic by default. Any deny is treated as primary to overtake the other rules that may be more specific below it. Let's define a simple security group, for example. This one's intended for EC2 instances hosting web applications. We'll allow HTTPS traffic from the world, and SSH traffic from inside private IPs. And this will allow us to expose a secure website to web users and lets anyone on our private network log into the machine via SSH. Of course, SSH users will still need a valid login to the machine. We'll keep outgoing rules at their default, fully open, so the machine can call out to version control repos, third-party APIs and the like. So here's our data center again. The security group we just defined should be attached to each instance in the public-facing web server subnet, like so. Because it allows both incoming and outgoing web traffic, it covers fully one, two, three, four, five of the rules we had before. If we define another security group that lets traffic flow from the web server's subnet, we can attach it to these database instances and get rid of the last three rules. Finally, there's no need for our expensive firewall appliance. The security groups act like local firewalls for each instance. What about this instance here? Is it vulnerable since there's no longer a border firewall? Not at all. Without a security group, AWS allows no traffic in or out. Advantages of security groups include reuse. You can reuse common sets of network rules. You can easily see where rule sets are applied. You keep network access rules close to the resource they control. And you can apply multiple groups to a single resource. Finally, my favorite feature of AWS Security Groups, they can be configured not only to allow traffic from certain IP ranges, they can also be configured to allow traffic from certain other security groups. Take the database example we described earlier. It could be configured to allow say, postgres traffic on port 5432 from any host in the web subnet range. On the other hand, look at our diagram. Every host the database needs to trust is using the web security group represented in orange. If the blue group could just trust all the oranges, we'd be in business. That way when we create a new web server and attach the web server security group to it, the orange one, we get network connectivity to our databases enabled for free. And we can do just that, allow traffic from another security group rather than an IP range. See this example here, where we provide the security group's ID as a target, rather than an IP address. This capability unlocks a whole new way of thinking about network security. In AWS, you can focus less on point-to-point rules in IP ranges and more on what kinds of resources should trust each other. Once you start doing this, a security group starts to look less like a mini firewall and more like a badge that grants you entry to secure places. Features like this really help AWS developers tackle traditional concepts like network security in exciting new ways.

HIROKO NAKAMURA - AWS for Non-Engineers

Book 1 AWS Certified Cloud Practitioner Exam

Use case: AWS Free Tier

So, now that we are armed with an AWS account and AWS Free Tier, let's explore a use case for a real project you can create using mostly free resources. Imagine you have to create and host a WordPress website using AWS. You can very quickly spin up an Elastic Compute Cloud or EC2 Instance, that comes loaded with WordPress. AWS has a marketplace for preconfigured servers called Amazon Machine Images or AMI. These are basically templates of servers that you can create and immediately get it preconfigured to a certain way. In this case, a company called Bitnami has created a WordPress AMI called WordPress Certified by Bitnami. It is Free Tier eligible and runs on an Ubuntu server. You will be led through the setup process and once you're through, you will have a WordPress website set up and ready to go. EC2s have fairly long URLs through, which could be something like ec2-52-204-122-132.compute-1.amazonaws.com. That's usually not a very attractive way to introduce your blog to your new friends. You would probably want something like mycoolblog.com to take your visitors to your brand new blog. AWS has a service to help you do just that. The simplest way is to purchase the domain name that you want using AWS's domain name registrar Route 53. A domain name registrar is like a phone book. To visit a website you input a domain name like mycoolblog.com and the DNS finds it in an online directory of IP addresses. It then sends your request to the appropriate server so you can load the website. By purchasing your domain on Route 53 and matching the domain name with the IP address of your EC2 Instance, you can make the address mycoolblog.com load your WordPress website. Now, Route 53 costs a few dollars a year for the domain registration and charges a separate monthly usage fee, however, the monthly usage fee for me is around 50 cents a month and domain registration itself was around $12, so for a whole year of website hosting the costs are fairly minimal. If you were thinking about starting a blog for cheap using AWS might just be the way to go, and it doesn't hurt that you are getting some hands-on experience with different services at AWS. There are many resources available on how to set one up ranging in complexity from a simple one, like what we just did using Route 53 and EC2, to using other services like CloudFront, AWS Certificate Manager and Elastic Load Balancer to help secure the website and make sure it stays up even if someone tried to take your blog down with a DDoS attack. Your creativity can take the reins to create just the project you were dreaming of with AWS Free Tier and other services.

Study break: Exam tips and resources

We began this course with what even is the cloud? Since then, we've come a long way. Let's get prepped for the exam. We've learned about the cloud, cloud computing, Amazon Web Services, and various but essential cloud computing concepts to help begin the preparation process for the AWS Certified Cloud Practitioner Exam. In this video, we will review major concepts you should know about for the AWS Certified Cloud Practitioner exam's Cloud Concepts domain. These topics are what is AWS? Six advantages of cloud computing, three cloud computing models, three cloud computing deployments, and five pillars of a well architected framework. We will also throw in some study tips for memorizing certain concepts in preparation for the exam, let's get started. The AWS Certified Cloud Practitioner exam is the most fundamental certification exam that AWS offers to help validate the candidates' overall fundamental understanding of the AWS cloud. It includes four domains which are cloud concepts security, technology, and billing and pricing. We began with cloud computing and then went over to the cloud concepts domain. For the cloud concepts domain part of the exam, AWS wants you to define the AWS cloud and its value proposition, identify aspects of AWS cloud economics and list the different cloud architecture design principles. AWS or Amazon Web Services is a cloud computing platform created by Amazon and currently holds the world's highest market share in the cloud computing sphere. It provides many different IT services on the cloud and helps to make it easier, faster and cheaper to run your IT infrastructure compared with legacy on-premises IT infrastructure. According to AWS, there are six distinct advantages of utilizing cloud computing over on-premises IT infrastructure. The advantages are trade capital expense for variable expense, benefit from massive economies of scale, stop guessing about capacity, increase speed and agility, stop spending money running and maintaining data centers and go global in minutes. Basically, utilizing cloud computing is faster, cheaper and more agile than utilizing your own data centers. There are three types of cloud computing models and three types of cloud computing deployments. The three types of cloud computing models are Software as a Service, SaaS, Infrastructure as a Service, IaaS and Platform as a Service, PaaS. The three types of cloud computing deployments are public cloud, hybrid cloud and private or on-premises cloud. Want a way to memorize them? Try out my silly memorization method, SIP PHO. SIP is an acronym for the three cloud computing models: software as a service, infrastructure as a service and platform as a service. And oh boy, do I love myself a good bowl of Pho, PHO stands for the cloud computing deployment models: public, hybrid and on-premises or private cloud. There are six pillars in a well architected framework. They provide best practices for specific areas of running an AWS cloud IT infrastructure. To help me memorize these six pillars, I created the acronym CROPSS with two S's. The pillars are cost optimization, reliability, operational excellence, performance efficiency, security and sustainability. What do you think? Do you think you'd be able to answer questions in regards to all of the topics we learned about in this course? If not, please, don't hesitate to go back and re-watch some of the videos and take notes.

Book 2 SECURITY