1. **Amazon Web Services (AWS)**

AWS Terminology

AWS IoT: AWS IoT is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices.

Certificate Manager: AWS Certificate Manager lets you easily provision, manage, and deploy Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services.

CloudFormation: AWS CloudFormation lets you create and update a collection of related AWS resources in a predictable fashion.

CloudFront: Amazon CloudFront provides a way to distribute content to end users with low latency and high data transfer speeds.

CloudSearch: AWS CloudSearch is a fully managed search service for websites and apps.

CloudTrail: AWS CloudTrail provides increased visibility into user activity by recording API calls made on your account.

Data Pipeline: AWS Data Pipeline is a lightweight orchestration service for periodic, data-driven workflows.

DMS: AWS Database Migration Service (DMS) helps you migrate databases to the cloud easily and securely while minimizing downtime.

DynamoDB: Amazon DynamoDB is a scalable NoSQL data store that manages distributed replicas of your data for high availability.

EC2: Amazon Elastic Compute Cloud (EC2) provides resizable compute capacity in the cloud.

EC2 Container Service: Amazon ECS allows you to easily run and manage Docker containers across a cluster of Amazon EC2 instances.

Elastic Beanstalk: AWS Elastic Beanstalk is an application container for deploying and managing applications.

ElastiCache: Amazon ElastiCache improves application performance by allowing you to retrieve information from an in-memory caching system.

Elastic File System: Amazon Elastic File System (Amazon EFS) is a file storage service for Amazon Elastic Compute Cloud (Amazon EC2) instances.

Elasticsearch Service: Amazon Elasticsearch Service is a managed service that makes it easy to deploy, operate, and scale Elasticsearch, a popular open-source search and analytics engine.

Elastic Transcoder: Amazon Elastic Transcoder lets you convert your media files in the cloud easily, at low cost, and at scale

Elastic MapReduce (EMR): Amazon Elastic MapReduce lets you perform big data tasks such as web indexing, data mining, and log file analysis.

Glacier: Amazon Glacier is a low-cost storage service that provides secure and durable storage for data archiving and backup.

IAM: AWS Identity and Access Management (IAM) lets you securely control access to AWS services and resources.

Inspector: Amazon Inspector enables you to analyze the behavior of the applications you run in AWS and helps you to identify potential security issues.

Kinesis: Amazon Kinesis services make it easy to work with real-time streaming data in the AWS cloud.

Lambda: AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.

Machine Learning: Amazon Machine Learning is a service that enables you to easily build smart applications.

OpsWorks: AWS OpsWorks is a DevOps platform for managing applications of any scale or complexity on the AWS cloud.

RDS: Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale familiar relational databases in the cloud.

Redshift: Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse that makes it cost-effective to analyze all your data using your existing business intell-igence tools.

Route 53: Amazon Route 53 is a scalable and highly available Domain Name System (DNS) and Domain Name Registration service.

SES: Amazon Simple Email Service (SES) enables you to send and receive email.

SNS: Amazon Simple Notification Service (SNS) lets you publish messages to subscribers or other applications.

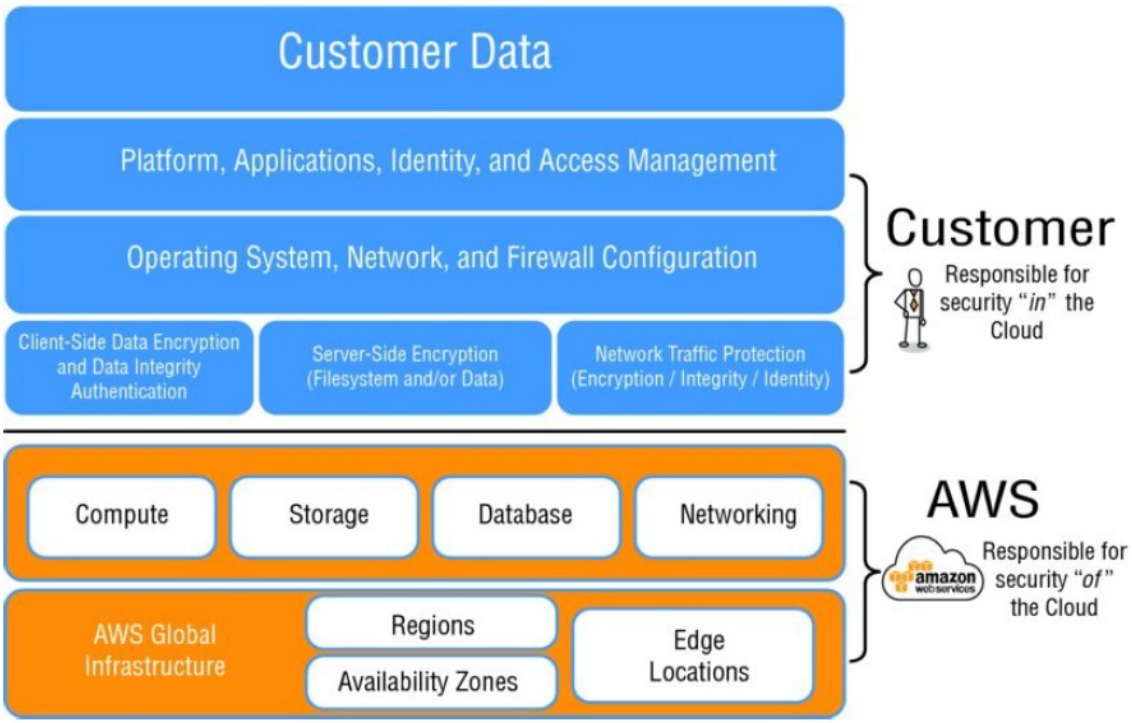
Storage Gateway: AWS Storage Gateway securely integrates on-premises IT environments with cloud storage for backup and disaster recovery.

SQS: Amazon Simple Queue Service (SQS) offers a reliable, highly scalable, hosted queue for storing messages.

SWF: Amazon Simple Workflow (SWF) coordinates all of the processing steps within an application.

S3: Amazon Simple Storage Service (S3) can be used to store and retrieve any amount of data.

VPC: Amazon Virtual Private Cloud (VPC) lets you launch AWS resources in a private, isolated cloud.



15. What are the popular DevOps tools?

Answer: In an AWS DevOps Engineer interview, this is the most common AWS interview questions for DevOps. To answer this question, mention the popular DevOps tools with the type of tool –

- Jenkins – Continuous Integration Tool
- Git – Version Control System Tool
- Nagios – Continuous Monitoring Tool
- Selenium – Continuous Testing Tool
- Docker – Containerization Tool
- Puppet, Chef, Ansible – Deployment and Configuration Management Tools

$aws s3 cp s3://cda-test-ver-2018/myfile.txt ./myfile.txt

$aws s3api list-object-versions –bucket cda-test-ver-2018

a. Elastic Compute Cloud (EC2)
b. Elastic Load Balancer (ELB)
c. Elastic Block Storage (EBS)
d. Simple Storage Service (S3)
e. CloudFront
f. Relational Database Service (RDS)
g. Virtual Private Cloud (VPC)
i. Identity & Access Management (IAM)
j. Cloud Trail
k. Cloud Watch
l. Elastic Cache
m. OpsWorks
n. Simple Notification Service (SNS)
o. Command Line Interface (CLI)
p. Lambda
q. Elastic File System (EFS)
r. Run Command
s. Auto Scaling
t. CloudFormation
u. EC2 Container Service (ECS)

Class A	10.0.0.0	10.255.255.255	10.0.0.0/8
Class B	172.16.0.0	172.31.255.255	172.16.0.0/12
Class C	192.168.0.0	192.168.255.255	192.168.0.0/16

192.168.1.0	192.168.1.255	192.168.1.0/24
32 - 24 = 8	2^8 = 256 IP address per subnet	**256 -5 = 251 Available**
Network ID	192.168.1.0	
VPC route / DHCP	192.168.1.1	
DNS	192.168.1.2	
Future Use	192.168.1.3	
Available to use	192.168.1.4	
Available to use	192.168.1.5	
Available to use	192.168.1.6	
Available to use	192.168.1.7	
Available to use	192.168.1.8	
Available to use	…	
Available to use	192.168.1.254	
Broadcast IP	192.168.1.255	

Type	Protocol	Port Rang	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	
RDP	TCP	3389	Anywhere	0.0.0.0/0	

1. **$curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"**
2. $ **python get-pip.py –user**
3. $ **pip install awscli –user**
4. **$ aws configure**
   - AWS Access Key ID [None]: *AKIAIOSFODNN7EXAMPLE*
   - AWS Secret Access Key [None]: *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*
   - Default region name [None]: *us-east-2*

- Default output format [None]: **json**

5. $ **aws ec2 describe-regions --output table**
6.

$ **aws ec2 create-security-group --group-name devenv-sg --vpc-id vpc-*xxxxxxxx* --description "security group for development environment"**

```
{
"GroupId": "sg-b018ced5"
}
```

$ **aws ec2 authorize-security-group-ingress --group-name devenv-sg --protocol tcp --port 22 --cidr 172.20.*0.0/16***

$ **aws ec2 create-key-pair --key-name devenv-key --query 'KeyMaterial' --output text > devenv-key.pem**

$ **chmod 400 devenv-key.pem**

$ **ssh -i devenv-key.pem *user*@*54.183.22.255***

- Create a VPC with a 10.0.0.0/16 CIDR block.

rou

aws ec2 create-vpc --cidr-block *172.20.0.0/16*

aws ec2 describe-vpcs --vpc-id ***vpc-2f09a348***

- Using the VPC ID from the previous step, create a subnet with a 10.0.1.0/24 CIDR block.

aws ec2 create-subnet --vpc-id *vpc-2f09a348* --cidr-block *172.20.1.0/24*

- Create a second subnet in your VPC with a 10.0.0.0/24 CIDR block.

aws ec2 create-subnet --vpc-id *vpc-2f09a348* --cidr-block *172.20.2.0/24*

- Create a third subnet in your VPC with a 10.0.0.0/24 CIDR block.

aws ec2 create-subnet --vpc-id *vpc-2f09a348* --cidr-block *172.20.3.0/24*

Make Your Subnet Public

After you've created the VPC and subnets, you can make one of the subnets a public subnet **by attaching an Internet gateway to your VPC**, creating a custom route table, and configuring routing for the subnet to the Internet gateway.

1. Create an Internet gateway.

aws ec2 create-internet-gateway

NOTE: In the output that's returned, take note of the Internet gateway ID.

2. Using the ID from the previous step, attach the Internet gateway to your VPC.

aws ec2 attach-internet-gateway --vpc-id *vpc-2f09a348* --internet-gatewayid *igw-1ff7a07b*

3. Create a custom route table for your VPC.

aws ec2 create-route-table --vpc-id *vpc-2f09a348*

In the output that's returned, take note of the route table ID.

4. Create a route in the route table that points all traffic (**0.0.0.0/0**) to the Internet gateway.

aws ec2 create-route --route-table-id *rtb-c1c8faa6* --destination-cidr-block *0.0.0.0/0* --gateway-id *igw-1ff7a07b*

5. To confirm that your route has been created and is active, you can describe the route table and view the results.

aws ec2 describe-route-tables --route-table-id *rtb-c1c8faa6*

6. **The route table is currently not associated with any subnet.** You need to associate it with a subnet in your VPC so that traffic from that subnet is routed to the Internet gateway. **First, use the describe-subnets command** to get **your subnet IDs**. You can use the **--filter option to return the subnets** for your new VPC only, and the --query option to return only the subnet IDs and their CIDR blocks.

aws ec2 describe-subnets --filters "Name=vpc-id,Values=*vpc-2f09a348*" --query 'Subnets[*].{ID:SubnetId,CIDR:CidrBlock}'

You can choose which subnet to associate with the custom route table, for example, **subnetb46032ec**.

This subnet will be your public subnet

aws ec2 associate-route-table --subnet-id *subnet-b46032ec* --route-table-id *rtbc1c8faa6*

You can optionally modify the public IP addressing behavior of your subnet so that an instance launched into the subnet automatically receives a public IP address.

Otherwise, **you should associate an Elastic IP address with your instance** after launch so that it's reachable from the Internet.

aws ec2 modify-subnet-attribute --subnet-id *subnet-b46032ec* --map-public-ip-on-launch

7. You can choose which subnet to associate with the custom route table, for example, **subnetb46032ec.** This subnet will be your public subnet.

aws ec2 associate-route-table --subnet-id *subnet-b46032ec* --route-table-id *rtbc1c8faa6*

8. You can **optionally modify the public IP addressing behavior of your subnet** so that an instance launched into the subnet automatically receives a public IP address.

Otherwise**, you should associate an Elastic IP address with your instance after launch** so that it's reachable from the Internet.

aws ec2 modify-subnet-attribute --subnet-id *subnet-b46032ec* --map-public-ip-on-launch

1. Create a key pair and use the --query option and the --output text option to pipe your private key directly into a file with the **.pem** extension.

aws ec2 create-key-pair --key-name *MyKeyPair* --query 'KeyMaterial' --output text > *MyKeyPair.pem*

2. Create a security group in your VPC, and add a rule that allows SSH access from anywhere.

aws ec2 create-security-group --group-name *SSHAccess* --description "*Security group for SSH access*" --vpc-id *vpc-2f09a348*

aws ec2 authorize-security-group-ingress --group-id *sg-e1fb8c9a* --protocol *tcp* --port *22* --cidr *0.0.0.0/0*

Note
If you use **0.0.0.0/0**, you enable **all IPv4 addresses to access your instance using SSH**. This is acceptable for this short exercise, but in production, authorize only a specific IP address or range of addresses.

Launch an instance into your public subnet, using the security group and key pair you've created. In the output, take note of the instance ID for your instance.

aws ec2 run-instances --image-id *ami-a4827dc9* --count 1 --instance-type *t2.micro* –keyname *MyKeyPair* --security-group-ids *sg-e1fb8c9a* --subnet-id *subnet-b46032ec*

In this example, the AMI is an Amazon Linux AMI in the US East (N. Virginia) region. If you're in a different region, you'll need the AMI ID for a suitable AMI in your region. For more information, see Finding a Linux AMI in the *Amazon EC2 User Guide for Linux Instances*

Your instance must be in the running state in order to connect to it. Describe your instance and confirm its state, and take note of its public IP address.

aws ec2 describe-instances --instance-id *i-0146854b7443af453*

When your instance is in the running state, you can connect to it using an SSH client on a Linux or Mac OS X computer by using the following command:

ssh -i "*MyKeyPair.pem*" *ec2-user@52.87.168.235*

Configure an Egress-Only Private Subnet

You can configure the second subnet in your VPC to be an IPv6 egress-only private subnet. Instances that are launched in this subnet are able to access the Internet over IPv6 (for example, to get software updates) through an egress-only Internet gateway, but hosts on the Internet cannot reach your instances.

Create an egress-only Internet gateway for your VPC. In the output that's returned, take note of the gateway ID.

aws ec2 create-egress-only-internet-gateway --vpc-id *vpc-2f09a348*

Create a custom route table for your VPC. In the output that's returned, take note of the route table ID.

aws ec2 associate-route-table --subnet-id *subnet-a46032fc* --route-table-id *rtb-abc123ab*

From your private instance, test that you can connect to the Internet by running the ping command for a website that has ICMP enabled, for example:

ping -n ietf.org

Cleanup

After you've verified that you can connect to your instance, you can terminate it if you no longer need it. To do this, use the terminate-instances command. To delete the other resources you've created in this example, use the following commands in their listed order:

1. Delete your security group:

aws ec2 delete-security-group --group-id *sg-e1fb8c9a*

2. Delete your subnets:

aws ec2 delete-subnet --subnet-id *subnet-b46032ec*

aws ec2 delete-subnet --subnet-id *subnet-a46032fc*

3. Delete your custom route table:

aws ec2 delete-route-table --route-table-id *rtb-c1c8faa6*

4. Detach your Internet gateway from your VPC:

aws ec2 detach-internet-gateway --internet-gateway-id *igw-1ff7a07b* --vpc-id *vpc-2f09a348*

5. Delete your Internet gateway:

aws ec2 delete-internet-gateway --internet-gateway-id *igw-1ff7a07b*

6. Delete your VPC:

aws ec2 delete-vpc --vpc-id *vpc-2f09a348*

Amazon Virtual Private Cloud (Amzaon VPC)

Amazon VPC is the networking layer for Amazon Elastic Compute Cloud (Amazon EC2) and it allows you to build your own virtual network with AWS.

- You should remember the following points about route tables:
- Your VPC has an implicit router.
- Your VPC automatically comes with a main route table that you can modify.
- You can create additional custom route tables for your VPC.
- Each subnet must be associated with a route table, which controls the routing for thesubnet. If you don't explicitly associate a subnet with a particular route table, the subnet uses the main route table.
- You can replace the main route table with a custom table that you've created so that each new subnet is automatically associated with

A VPC consists of the following components:

- Subnets
- Route tables
- DHCP option sets
- Security groups
- Network ACLs

optional components:

- IGWs
- EIP addresses
- Endpoints
- Peering
- NAT instance and NAT gateway
- VPG, CGW, and VPN

-

configure the following values within a DHCP option set:

- ***domain-name-servers***—The IP addresses of up to four domain name servers, separated by commas. The default is AmazonProvidedDNS.
- ***domain-name***—Specify the desired domain name here (for example, cloudcybersafe.com).
- ***ntp-servers***—The IP addresses of up to four Network Time Protocol (NTP) servers, separated by commas
- ***netbios-name-servers***—The IP addresses of up to four NetBIOS name servers,
- separated by commas
- ***netbios-node-type***—Set this value to 2.

  Every Amazon VPC must have only one DHCP option set assigned to it.

- You must first allocate an EIP for use within a VPC and then assign it to an instance.
- **EIPs are specific to a region (that is, an EIP in one region cannot be assigned to an instance within an Amazon VPC in a different region).**
- There is a one-to-one relationship between network interfaces and EIPs.
- You can move EIPs from one instance to another, either in the same Amazon VPC or a different Amazon VPC within the same region.
- EIPs remain associated with your AWS account until you explicitly release them.
- There are charges for EIPs allocated to your account, even when they are not associated with a resource.

You must do the following to create a public subnet with Internet access:

- Attach an IGW to your Amazon VPC.
- Create a subnet route table rule to send **all non-local traffic (0.0.0.0/0)** to the IGW.
- Configure your **network ACL**s and **security group rules** to allow relevant traffic to flow to and from your instance.
- instance to send and receive traffic from the Internet: Assign a public IP address or EIP address.

Endpoints

An Amazon VPC *endpoint* enables you to create a private connection between your Amazon VPC and another AWS service **without requiring access over the Internet** or through a NAT instance, VPN connection, or AWS Direct Connect

You must do the following to create an Amazon VPC endpoint:

- Specify the Amazon VPC.
- Specify the service. **A service is identified by a prefix list of the form com.amazonaws.<region>.<service>.**

- Specify the policy. You can allow full access or create a custom policy. This policy can be changed at any time.
- Specify the route tables. A route will be added to each specified route table, which will state the service as the destination and the endpoint as the target.

Security Groups

A *security group* is a virtual stateful firewall that controls inbound and outbound network traffic to AWS resources and Amazon EC2 instances. All Amazon EC2 instances must be launched into a security group. If a security group is not specified at launch, then the instance will be launched into the default security group for the Amazon VPC. **The default security group allows communication between all resources within the security group, allows all outbound traffic, and denies all other traffic**.

- You can create up to **500 security groups** for each Amazon VPC.
- You can add up **to 50 inbound and 50 outbound rules** to each security group. If you need to apply more than 100 rules to an instance, you can associate up to five security groups with each network interface.
- **You can specify allow rules, but not deny rules. This is an important difference between security groups and ACLs.**
- You can specify separate rules for inbound and outbound traffic.
- **By default, no inbound traffic is allowed until you add inbound rules to the security group.**
- By default, new security groups have an outbound rule that allows all outbound traffic.
- You can remove the rule and add outbound rules that allow specific outbound trafficonly.
- **Security groups are stateful. This means that responses to allowed inbound traffic are allowed to flow outbound regardless of outbound rules and vice versa**. This is an important difference between security groups and network ACLs.
- Instances associated with the same security group can't talk to each other unless you add rules allowing it (with the exception being the default security group).
- You can change the security groups with which an instance is associated after launch, and the changes will take effect immediately

Network Access Control Lists (ACLs)

A network *access control list (ACL)* is another layer of security that **acts as a stateless firewall on a subnet level.** A network ACL is a numbered list of rules that AWS evaluates in order, **starting with the lowest numbered rule,** to determine whether traffic is allowed in or out of any subnet associated with the network ACL. Amazon VPCs are created with a modifiable default network ACL associated with every subnet that allows all inbound and outbound traffic. When you create a custom network ACL, its

initial configuration will deny all inbound and outbound traffic until you create rules that allow otherwise.

Network Address Translation (NAT) Instances and NAT Gateways
By default, any instance that you launch into a private subnet in an Amazon VPC is not able to communicate with the Internet through the IGW. AWS provides NAT instances and NAT gateways to allow instances deployed in private subnets to gain Internet access.

NAT Gateway
A *NAT gateway* is an Amazon managed resource that is designed to operate just like a NAT instance, but it is simpler to manage and highly available within an Availability Zone.

NAT Gateway
A *NAT gateway* is an Amazon managed resource that is designed to operate just like a NAT instance, but it is simpler to manage and highly available within an Availability one.
NAT Instance
A *network address translation (NAT) instance* is an Amazon Linux Amazon Machine Image (AMI) that is designed to accept traffic from instances within a private subnet, translate the source IP address to the public IP address of the NAT instance, and forward the traffic to the IGW.

The Elastic Load Balancing service allows you to **distribute traffic across a group of Amazon EC2 instances in one or more *Availability Zones*,** enabling you to achieve high availability in your applications. Elastic Load Balancing supports routing and load balancing of Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), *Transmission Control Protocol (TCP)*, and *Secure Sockets Layer (SSL)* traffic to Amazon EC2 instances. Elastic Load Balancing provides a stable, single *Canonical Name record (CNAME)* entry point for *Domain Name System (DNS)* configuration and supports both Internet-facing and internal application-facing load balancers. Elastic Load Balancing supports health checks for Amazon EC2 instances to ensure traffic is not routed to unhealthy or failing instances. Elastic Load Balancing can automatically scale based on collected metrics

Elastic Load Balancing is a highly available service itself and can be used to help build highly available architectures An AWS recommended best practice is always to reference a load balancer by its DNS name, instead of by the IP address of the load balancer, in order to provide a single, stable entry point. To ensure that the load balancer is responsible for closing the connections to your back-end instance, make sure that the value you set for the keep-alive time is greater than the idle timeout setting on your load balancer

Amazon Cloud Watch

Amazon CloudWatch is a monitoring service for AWS Cloud resource and the application running on AWS. It allows organization to collect and track metrics, collect and monitors logs files, and set alarms.

- Monitor your AWS resource and your application in real time.
- Amazon CloudWatch offers either basic or detailed monitoring for supported AWS products.
- Basic monitoring send data points to Amazon CloudWatch every five minutes for a limited number of preselected metrics at no charge.
- Detailed monitoring sends data points to Amazon CloudWatch every minute and allows data aggregation for an additional charge.
- If you want to use detailed monitoring, you must enable it basic is the default.
- Amazon CloudWatch supports multiple type of action such as sending notification to an Amazon Simple Notification Service (SNS) or executing an Auto Scaling policy.
- Amazon CloudWatch does not aggregate data across regions but can aggregate across Availability Zones within a region.
- Each AWS account is limited to 5,000 alarms per AWS account, and metrics data is retained for two weeks by default

Auto Scaling

Scale out quickly; scale in slowly.

A distinct advantage of deploying applications to the cloud is the ability to launch and then release servers in response to variable workloads. Provisioning servers on demand and then releasing them when they are no longer needed can provide significant cost savings for workloads that are not steady state.

Steady state workloads that need a consistent number of Amazon EC2 instances at all times can use Auto Scaling to monitor and keep that specific number of Amazon EC2 instances running.

Manual scaling out can be very useful to increase resources for an infrequent event, such as the release of a new game version that will be available for download and require a user registration. For extremely large-scale events, even the Elastic Load Balancing load balancers can be pre-warmed by working with your local solutions architect or AWS Support.

Recurring events such as end-of-month, quarter, or year processing, or scheduled and recurring automated load and performance testing, can be anticipated and Auto Scaling can be ramped up appropriately at the time of the scheduled event

In large deployments of Amazon EC2 instances, Auto Scaling can be used to make rolling out a patch to your instances easy. The launch configuration associated with the Auto Scaling group may be modified to reference a new AMI and even a new Amazon EC2 instance if needed. Then you can deregister or terminate instances one at a time or in small groups, and the new Amazon EC2 instances will reference the new patched AMI

- Auto Scaling has several schemas palns 1) Maintain Current Instance levels, 2) Manual Scaling,
- A launch configuration is the template that Auto Scaling uses to create new instance, and it is compoes of the configuration name, Amazon Machine Image (AMI), Amazon EC2 Instance type, security group, and instance key pair.
- An Auto Scaling group is collection of Amazon EC2s instances managed by the Auto Scaling service.

- $aws autoscaling describe-account-limits
- $aws autoscaling create-auto-scaling-group –auto-scaling-group-name myASG –launch-configuration-name myLC –availabiltiiy-zones us-east-1a, us-east-1c –min-size 5 –max-size 20 –desirect-capacity 3 –load-balancer-name myELB.
- A recommended best practice is to SCALE out quickly and SCALE IN slowly so you can respond to burst or spkike.
- Scale out quickly; Scale in slowly.
- Instance that are more stateless instead of stateful will more gracefully enter and exit and Auto Scaling group.

- ❖ Elastic Load Balancing, which is used to distribute traffic across a group of Amazon EC2 instances in one or more Availability Zones to achieve greater levels of fault tolerance for your applications.
- ❖ Amazon CloudWatch, which monitors resources and applications. Amazon CloudWatch is used to collect and track metrics, create alarms that send notifications, and make changes to resources being monitored based on rules you define.
- ❖ Auto Scaling, which allows you to automatically scale your Amazon EC2 capacity out and in using criteria that you define.

These three services can be used very effectively together to create a highly available application with a resilient architecture on AWS.

A principal is an IAM entity that is allowed to interact with AWS resources. A principal can be permanet or tempory, and it can represent a human or an application.

AWS Identity and Access Management (IAM)

IAM is a powerful service that allows you to control how people and programs are allowed to manipulate your AWS infrastructure. IAM uses traditional identity concepts such as users, groups, and access control policies to control who can use your AWS account, what services and resources they can use, and how they can use them.

First, IAM is not an identity store/authorization system for your applications. The permissions that you assign are permissions to manipulate AWS infrastructure, not permissions within your application.

Second, IAM is not operating system identity management. Remember that under the shared responsibility model, you are in control of your operating system console and configuration. Whatever mechanism you currently use to control access to your server infrastructure will continue to work on Amazon Elastic Compute Cloud (Amazon EC2) instances, whether that is managing individual machine login accounts or a directory service such as Active Directory or Lightweight Directory Access Protocol (LDAP). You can run an Active Directory or LDAP server on Amazon EC2, or you can extend your onpremises system into the cloud. AWS Directory Service will also work well to provide Active Directory functionality in the cloud as a service, whether standalone or integrated with your existing Active Directory.

THE AWS

Root User When you first create an AWS account, you begin with only a single sign-in principal that has complete access to all AWS Cloud services and resources in the account.

A good first step is to use the root user to create a **new IAM group called "IAM Administrators"** and **assign the managed policy, "IAMFullAccess."** Then create a new IAM user called "**Administrator," assign a password**, and add it to the **IAM Administrators group.** At this point, you can log off as the root user and perform all further administration with the IAM user account.

- IAM User (Users are an excellent way to enfor the principle of least privilege;
- Roles/Temporary Security Tokens; Roles are used to grant specific privileges to specific actor for a set duration of time.
- The range of a temporary security toke lifetime is 15 minutes to 36 hours.
- Roles and temporary security tokes enable a number of use cases:
    1. Amazon EC2 Roles

2. Cross-Accoutn Access
3. Federation

- IAM can integrate with two different types of outside Identity Provider (IdP) . OpenID Connect and Security Assertion Markup Language 2.0 (SAML)
- What actions a principle can and cannot perform is called authorization.

Roles/Temporary Security Tokens
Roles and temporary security tokens are very important for advanced IAM usage, but many AWS users find them confusing. Roles are used to grant specific privileges to specific actors for a set duration of time. These actors can be authenticated by AWS or some trusted external system. When one of these actors assumes a role, AWS provides the actor with a temporary security token from the *AWS Security Token Service (STS)* that the actor can use to access\

The range of a temporary security token lifetime is 15 minutes to 36 hours.

Using IAM roles for Amazon EC2 removes the need to store AWS credentials in a configuration file.

Multi-Factor Authentication requires you to verify your identity with both something you *know* and something you *have.*

Using predefined managed policies ensures that when new permissions are added for new features, your users will still have the correct access.

Rotating Keys
The security risk of any credential increases with the age of the credential. To this end, it is a security best practice to *rotate access keys* associated with your IAM users. IAM facilitates this process by allowing two active access keys at a time. The process to rotate keys can be conducted via the console, CLI, or SDKs:

1. Create a new access key for the user.
2. Reconfigure all applications to use the new access key.
3. Disable the original access key (disabling instead of deleting at this stage is critical, as it allows rollback to the original key if there are issues with the rotation).
4. Verify the operation of all applications.
5. Delete the original access key.

Access keys should be rotated on a regular schedule.

It is important to know how conflicts between these permissions are resolved:
1. Initially the request is denied by default.

2. All the appropriate policies are evaluated; **if there is an explicit "deny" found in any policy, the request is denied and evaluation stops.**
3. If no explicit "deny" is found and an explicit "allow" is found in any policy, the request is allowed.
4. If there are no explicit "allow" or "deny" permissions found, then the default "deny" is maintained and the request is denied.

Resolving Multiple

Amazon's Managed database Service		
Amazon RDS (RDBMS)	**Amazon DynamoDB (NOSQL)**	**Amazon RedShift (Data warehouses)**

Operational Responsibilities Customer vs AWS			
	Database On-Perm	**Database on EC2 instance**	**Database on Amazon RDS**
App Optimization	Customer	Customer	Customer
Scaling	Customer	Customer	AWS
High Availability	Customer	Customer	AWS
Backups	Customer	Customer	AWS
DB Engine Patches	Customer	Customer	AWS
Software Installation	Customer	Customer	AWS
OS Installation	Customer	AWS	AWS
Server maintenance	Customer	AWS	AWS
Rack and Stack Data Center	Customer	AWS	AWS

Database systems and engines can be grouped into two broad categories	
Relational Database Management Systems (RDBMS)	Non-Relational Database (NoSQL)
	Use case • Managing user session state • User profiles • Shopping card data • Time-series data •
MySQL (Open source)	Amazon DynamoDB

MariaDB (Open Source) built by the creators of MySQL	Hbase
PostgreSQL (Open source)	MongoDB
Microsoft SQL Server	Cassandra
Oracle	CouchDB
Amazon Aurora (Open Source) the internal components of MySQL to take more service-oriented approach.	Riak

Recap

- Amazon RDS is a managed relational database in the cloud
- Amazon Aurora
- MySQL
- MariaDB
- Oracle
- SQL Server
- PostgreSQL

- Managed service -> user not required to manage OS
- No access provided to the underlying OS.
- Can provision or resize resources on demand
- Provided with an endpoint, which you can connect using standard DB connection tools/libraries
- Read Replica

Amazon RDS MySQL allows you to connect using standard MySQL tools such as MySQL Workbench or SQL Workbench/J. Amazon RDS MySQL supports *Multi-AZ* deployments for high availability and ***read replicas*** **for horizontal scaling**.

Amazon RDS PostgreSQL also supports Multi-AZ deployment for high availability and **read replicas for horizontal scaling.**

AWS supports MariaDB version 10.0.17. Amazon RDS fully supports the XtraDB storage engine for MariaDB DB Instances and, like Amazon RDS MySQL and PostgreSQL, has support for Multi-AZ deployment and **read replicas**.

Amazon RDS Oracle supports three different editions of the popular database engine: Standard Edition One, Standard Edition, and Enterprise Edition

Edition	Encryption	Multiple Availability Zone
Standard One	KMS	Yes
Standard	KMS	Yes
Enterprise	KMS and TDE	Yes
Amazon RDS supports Transparent Data Encryption (TDE) for SQL Server (SQL Server Enterprise Edition) and Oracle (part of the Oracle Advanced Security option available in Oracle Enterprise Edition). **The TDE feature automatically encrypts data before it is written to storage and automatically decrypts data when it is read from storage**. If you require your MySQL data to be encrypted while at rest in the database, your application must manage the encryption and decryption of data.		
NOTE: **AWS Key Management Service (AWS KMS)** *AWS KMS* is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data. **AWS KMS lets you create keys that can never be exported from the service and that can be used to encrypt and decrypt data based on policies you define**.		

Amazon RDS Microsoft SQL Server also supports four different editions of SQL Server: Express Edition, Web Edition, Standard Edition, and Enterprise Edition

Edition	Encryption	Multiple Availability Zone
Express	KMS	
Web	KMS	
Standard	KMS	Yes
Enterprise	KMS and TDE	Yes
Amazon RDS supports Transparent Data Encryption (TDE) for SQL Server (SQL Server Enterprise Edition) and Oracle (part of the Oracle Advanced Security option available in Oracle Enterprise Edition). **The TDE feature automatically encrypts data before it is written to storage and automatically decrypts data when it is read from storage**. If you require your MySQL data to be encrypted while at rest in the database, your application must manage the encryption and decryption of data.		

Amazon Aurora

Amazon Aurora offers enterprise-grade commercial database technology Amazon Aurora is a fully managed service, is MySQL compatible out of the box, and provides for increased reliability and performance over standard MySQL deployments.

Aurora

- Delivers upto 5 time throughput of standard MySQL running on the same hardware
- Amazon Aurora increase MySQL performance and availabity
- Fault-tolerant and Selfhelign
- In case the who instance fails, Aurora will automatically failover to one of up to 15 read replicas

DynamoDB

- Fast scalable and fully managed NoSQL database service
- Key-Value and document store
- Table – Collection of data
- Items – individual entries in your table (rows)
- Attributes – the properties associated with the entries (columns)
- Dynamo uses SSDs to store data

Redshift (data ware house)

- Massively parallel processing (MPP)
- Fast execution of the most complex queries operating on large amounts of data
- Columnar data storage

Elasticache

Redis

- A fast, open source, in-memory data store and cache
- Both single-node and up to 15-shard clusters are available, enabling scalability to up to 3.55 TiB of in-memory data
- Use case such as Web, Mobile Apps, Gaming, Ad-Tech and IoT
-

Memcached

- A widely adopted memory object caching system
- ElasticCache is protocol complian with Memcached, so popular tools that
- You use today with existing Memcached enviroments will work
- Seamlessy with the service.

Route 53

- Domain Name System (DNS) translates human friendly domain names to an ip address (www.example.com to 10.1.0.20)
- Globally distributed service
- The Internet's "PHONE BOOK"
- IP addresses are used by machines to connect and communicate with one another
- 2 types IPv4 (32 bits) and IPv6 (128 bits)

What is Route 53

- Highly available and scalable Domain Name System web service
- Health-checking web service
- Public Hosted Zone - Container that holds information about how it should route traffic on the Internet for a domain and its subdomain
- Private Hosted Zone – with VPCs
- Resource Record Sets – tell the (DNS) how to route traffic for your domain
- Traffic Policies – complex routing configuration
- Health Check – monitor the health and performance of your web applications, web servers, and other resources
- Domain Registrations and Transfers
- Alias Records
- Provide an Amazon Route 53-specific extension to DNS
- Alias resource record set contains a pointer to CloudFront distributions, S3 buckets, Elastic Beanstalk, ELB or another Amazon Route 53 resource record set in the same hosted zone.

---- AWSCSA-23.3: Routing Policies

Simple: This is the default. Most commonly used when you have a single resource that performs a given function for your domain eg. one web server that serves content for the a website.

Think of one EC2 instance.

Weighted: Example you can send a percentage of users to be directed to one region, and the rest to others. Or split to different ELBs etc.

Used for splitting traffic regionally or if you want to do some A/B testing on a new website.

Latency: This is based on the lowest network latency for your end user (routing to the best region). You create a latency resource set for each region.

Route53 will select the latency resource set for the region that will give them the best result and repond with be resource set.

User -> DNS -> the better latency for an EC2 instance
Failover: When you want to create an active/passive set up. Route53 will monitor the health of your primary site using a health check.

Geolocation: Sending the user somewhere based on the location of the user.

Remember!

Can create an Alias record at the zone apex of a Domain, but cannot create a CNAME for it

CNAME queries will be charged

What are the Route 53 Policy Types?

1. Simple Routing
2. Weighted Routing
3. Latency Routing
4. Failover Routing
5. Geolocation Routing

Simple Routing

- When you have a single resource that perform a specific function for your domain
- Route 53 responds to DNS queries based only on the value in the resource record set (for example, the IP address in an A record)
- Default policy when you create a Record Set

Weighted Routing

- When you have multiple resource that perform the same function (for example, web servers that serve the same website) and you want to route traffic to them in proportion that you specify (for example, ¾ traffic to first site and 1/34 to second one)
- Use cases include – load balancing and testing new version of software (A/B testing)

Latency Routing

- When you have resource in multiple regions that perform the same function and you want to Route 53 to responds to DNS queries with the resource with the lowest latency for the users
- Latency resource records allowed for EC2 instance and ELBs.
- Latency between hosts on the Internet can change over time as a result of changes in network connectivity and routing.

Failover Routing

- When you want to configure active-passive failover, in which traffic is directed to one resource when it's available (primary) and to the other resource when the primary is not available (secondary)
- Useful for Disaster Recvery (DR)
- Route53 monitor your primary site using a health check.

Geolocation Routing

- When you want Route53 to respond to DNS queries based on the location of your users.
- Use Case include:
    - Restrict distribution of content to only the location in which you have distribution rights

- Localize your contend and present come or all your website in language of your users.

DNS is used to convert human friendly domain names into an Internet Protocol address (IP).

- DNS is like the phonebook. If someone wants to call you at your new house or location, they might lookup you up by name in the phonebook.
-

IP addresses come in 2 different forms: IPv4 and IPv6.

IPv4 is 32-bit. IPv6 is 128 bits.

IPv4 is used, but IPv6 is for the future.

DNS uses a hierarchical name structure, and different levels in the hierarchy are each septeated with a dot(.) Example www.amzaon.com or aws.amazonj.com In both cases COM is the Top Level Domain (TLD)

Top Level Domains

- .com
- .edu
- Etc

- ✓ Amazon Route 53 is an *authoritative DNS system*. An authoritative DNS system provides an update mechanism that developers use to manage their public DNS names. It then answers DNS queries, translating domain names into IP addresses so that computers can communicate with each other
- ✓ A *Top-Level Domain (TLD)* is the most general part of the domain. The TLD is the farthest portion to the right (as separated by a dot). Common TLDs are .com, .net, .org, .gov, .edu, and .io.
- ✓ Certain parties are given management control over TLDs by the Internet Corporation for Assigned Names and Numbers (ICANN). These parties can then distribute domain names under the TLD, usually through a domain registrar. These domains are registered with the **Network Information Center (InterNIC)**, a service of ICANN, which enforces the uniqueness of domain names across the Internet. Each domain name becomes registered in a central database, known as the WhoIS database
- ✓ A *domain name* is the human-friendly name that we are used to associating with an Internet resource. For instance, amazon.com is a domain name. Some people

will say that the amazon portion is the domain, but we can generally refer to the combined form as the domain name

- ✓ For example, 111.222.111.222 could be a valid IPv4 IP address. With DNS, we map a name to that address so that you do not have to remember a complicated set of numbers for each place you want to visit on a network.
- ✓ A *zone file* is a simple text file that contains the mappings between domain names and IP addresses. This is how a DNS server finally identifies which IP address should be contacted when a user requests a certain domain name.
- ✓ Zone files reside in name servers and generally define the resources available under a specific domain, or the place where one can go to get that information.

Domain Registrars

Names are registered with InterNIC - a service of ICANN. They enforce the uniqueness.

Route53 isn't free, but domain registrars include things like GoDaddy.com etc.

SOA Records

The record stores info about:

- supplies name of the server
- admin of the zone
- current version of the data file
- number of seconds a server should wait before retrying a failed zone
- Default number of seconds for TTL on resource records

NS Records

Name Server Records

- used by Top Level Domains to direct traffic to the Content DNS servers which contains the authoritative DNS records.

A Records

Address record - used to translate from a domain name to the IP address. A records are always IPv4. IPv6 is AAA.

TTL

Length that the DNS is cached on either the Resolving Server or on your PC. This is important from an architectural point of view.

CNAMES

Canonical Name (CName) can be used to resolve one domain name to another. eg. you may have a mobile website **m.example.com** that is used for when users browse to your domain on a mobile. You may also want **mobile.example.com** to point there as well.

Alias Records

Used to map resource record sets in your hosted zone to Elastic Load Balancers, CloudFront Distribution, or S3 buckets that are configured as websites.

Alias records work like a CNAME record in that you can map one DNS name (**www.example.com**) to another 'target' DNS name (**aeijrioea.elb.amazonaws.com**)

***Key Difference* - A CNAME can't be used for naked domain names (zone apex). You can't have a CNAME for acloud.guru. It must be either an A record or an Alias.**

The naked domain name MUST always be an A record, not a C name. eg dennis.com.

The Alias will map this A record to an ELB.

Summary

For an ELB, you need a DNS name to resolve to an ELB. You will always need an IPv4 domain to resolve this... which is why you have the Alias Record.

Records with alias records won't have you charged, whereas CName will.

---- AWSCSA-23.2: Creating a Route 53 Zone

Going from Route 53 to a load balancer to an EC2 Instance.

In EC2, launch an instance.

Bootstrap Script

Going from Route 53 to a load balancer to an EC2 Instance.

In EC2, launch an instance.

Bootstrap Script

Amazon Simple Queue Service (SQS)

- Simple Queue Service is a fully-managed message queuing service.
- Amazon SQS ensures delivery of each message at least once and supports multiple readers and writer interacting with the same queue.
- The service **does not guarantee First IN, First Out (FIFO)** delivery of message. For many distributed applications, each message can stand on its own and if all messages are delivered, the order is not important.
- Multiple producers and consumers can interact with SQL at the same time.
- Message size max 256 KB of text data (for example, JSON, XML, unformatted text)
- But billed at 64 KB chunks -> 256 KB billed as 4 request
- Nearly unlimited throughput
- At-Lest-Once Delivery
- Design application to be idempotent
- If you need order maintained, add sequencing info in each message so you can reorder them once they are received.
- Can send/receivd/delte a maximum of 10 message in request
- Message Retention Period – 1 minute – 14 days (default 4 days)
- Delay queues allow you to postpone the delivery of new messages in queue for specific number of seconds.
- To create aa delay queue, use CreateQueue and set the DelaySeconds attribute to any value between 0 to 900 (15minutes)
- The default value for DelaySeconds is 0.
- When a message is in the queue but is neither delayed nor in a visibility timeout, it is considered to be “IN FLIGHT”.
- You can have up to 120,000 messages IN FLIGHT at any given time .
- Amazon SQS supports up to 12 hours maximum visibility timeout.
- Your message are identified via a globally unique ID tat Amazon SQS returns.
- Amazon SQS uses three identifiers: 1) queue URL, 2) messageIDs 3) Receipt handles.
- Each message can have up to 10 attributes.
-

Amazon Simple Notification Service (SNS)

- Amazon SNS consist of two types of clients: publishers and subscribers (sometimes know as producers and consumers).
- Fast, flexible, fully managed push notification service
- Follow the “Pub/Sub” messaging paradigm

- Each topic has a unique name that identifies that Amazon SNS endpoints where publishers post messages and subscribers register for notifications.
- Send notification to Apple, Google, Fire OS, and Windows devices, as well as to Android devices in China with Baidu Cloud Push
- Create Simple Notification Service (SNS) topics and subscribe clients (subscribers) to them
- Amazon SNS can support a wide variety of needs, **including monitoring applications, workflow systems, time-sensitive information updates, mobile applications, and any other application that generates or consumes notifications**. For example, you can use Amazon SNS to relay events in workflow systems among distributed computer applications, move data
- A fanout scenario is when an Amazon SNS message is sent to a topic and then replicated and pushed to multiple Amazon SQS queues, HTTP endpoints, or email addresses This allows for parallel asynchronous processing.
- between data stores, or update records in business systems
- Can customize messages by protocol
- Flexible message delivery over multiple transport protocols:
    - HTTP
    - HTTPS
    - Email
    - Email-JSON
    - Simple Queue Service (SQS)
- Push email and text messaging are two ways to transmit messages to individuals or groups via email and/or SMS. For example, you can use Amazon SNS to push targeted news headlines to subscribers by email or SMS.
- Mobile push notifications enable you to send messages directly to mobile applications.

Amazon Simple Workflow Service (SWF)

- Simple WorkFlow is a managed web service for task coordination and state tracking across distributed application components
- SWF work with both application in the cloud as well as on-premise.
- A workflow can also involve human action
- Simple Workflow Service (SWF) redundantly store the tasks, reliably dispatched them to the application components, track their progress, and keeps their latest state

- Task will be performed at most once (task won't be duplicated).
- Actors can be workflow starters, deciders or activity workers. These actor communicate with Amazon SWF through its API.
- The logic that coordinates the tasks in a workflow is called the DECIDER.
- An ACTIVITY worker is a single computer process (or thread) that performs the activity in your workflow.
- Amazon SWF provides activity workers and decider with work assignments, given as one of three types of tasks: activity tasks, AWS Lambda tasks, and decisions tasks.
- Amazon SWF schedules a decision task when the workflow starts and whenever the stat of workflow changes, such as when an activity task completes.
- Task lists provide a flexible mechanism to route task to workers as you use necessitates.

Domain

- Contains and runs the desired workflow(s)

Worker

- Program that executes a workflow activity and provides the results back to SWF
- Task performed by human - > activity worker software used in the process.

Tasks

- Activity task – assigned to worker (for example, validating an order)

Decider task

- Tells the decider a change in the workflow execution has occurred
- The decider can then specify the next action

Amazon Simple Workflow Service (SWF) vs Amazon Simple Queue Service (SQS)

- Used to create decouple, distributed application
- SWF guarantees execution order, people can perform part of a workflow

- SQL offers best effort ordering
- SWF will assign a task only once (no duplicates)
- SQS can introduce duplicated (at least once delivery)
- SWF task can last up to 1 year
- SQS messages can be retained in the queue for max 14 days.

Exam Tips

- Route 53 + S3 (Static website hosting)
- The bucket must have the same name as your domain or subdomain (for example, if you want to use the subdomain shah.example.com, the name of the bucket must be shah.example.com)
- If you wish to route traffic for multiple domains like example.com and www.example.com to one bucket, you can create a bucket for each domain and configure all but one of the buckets to redirect traffic to the remaining bucket.

    - AWS **OpsWorks** is a configuration management service that uses Chef, an automation platform that treats server configuration as code.
    - **OpsWorks** uses Chef to automate how servers are configured, deployed, and managed across your EC2 instances or on-premises compute environment
    - (**Chef) Recipes -> Think of OpsWorks**
    - AWS OpsWorks uses the Chef framework you can bring your own recipes or leverage hundreds of community-build configuration.

CloudFormaiton

- AWS CloudFormaiton is a service that help you model and setup your AWS resource that your can spend less time managing those resource.
- Infrastructure as code
- CloudFormaiton is free
- You pay for the resources created by CloudFormaiton
- Stack – Collection of resource you want to be deployed together as a group
- JSON and YAML compatible
- AWS CloudFormaiton templates to define your AWS resource and their properties. A template is a text files whose format compile with the JSON standard.

- When you use AWS CloudFormation, you can reuse your template to set up your resources consistently and repeatedly. Just describe your resources once, and then provision the same resources over and over in multiple regions.
- When you use AWS CloudFormation, you manage related resources as a single unit called a stack. You create, update, and delete a collection of resources by creating, updating, and deleting stacks. All of the resources in a stack are defined by the stack's AWS CloudFormation template.
- Often you will need to launch stacks from the same template, but with minor variations, such as within a different Amazon VPC or using AMIs from a different region. These variations can be addressed using parameters. You can use parameters to customize aspects of your template at runtime, when the stack is built. For example, you can pass the Amazon RDS database size, Amazon EC2 instance types, database, and web server port numbers to AWS CloudFormation when you create a stack.
- Because environments are dynamic in nature, you inevitably will need to update your stack's resources from time to time. There is no need to create a new stack and delete the old one; you can simply modify the existing stack's template. To update a stack, create a *change set* by submitting a modified version of the original stack template, different input parameter values, or both.
- **If you want to delete a stack but still retain some resources in that stack, you can use a deletion policy to retain those resources. If a resource has no deletion policy, AWS CloudFormation deletes the resource by default**
- **Use Case**
  - ✓ **Quickly Launch New Test Environments**
  - ✓ **Reliably Replicate Configuration Between Environments**
  - ✓ **Launch Applications in New AWS Regions**

Templates can include maximum of 9 main sections

1. AWSTemplateFormatVersion –template version you wish to use
2. Descrition – text describing the template, must always follow the format version
3. Metadata – object that provide more into about the template
4. Paramters – values you can pass into your template at runtime
5. Mappying –mapping of keys and associated values.

AWS Elastic Beanstalk

AWS Elastic Beanstalk is the fastest and simplest way to get an application up and running on AWS. Developers can simply upload their application code, and the service automatically handles all of the details, such as resource provisioning, load balancing, Auto Scaling, and monitoring.

- ✓ There are key components that comprise AWS Elastic Beanstalk and work together to provide the necessary services to deploy and manage applications easily in the cloud. An *AWS Elastic Beanstalk application* is the logical collection

of these AWS Elastic Beanstalk components, which includes environments, versions, and environment configurations. In AWS Elastic Beanstalk, an application is conceptually similar to a folder

- An *environment* is an application version that is deployed onto AWS resources. Each environment runs only a single application version at a time; however, the same version or different versions can run in as many environments at the same time as needed. When an environment is created, AWS Elastic Beanstalk provisions the resources needed to run the application version that is specified
- At the time of this writing, AWS Elastic Beanstalk provides platform support for the programming languages Java, Node.js, PHP, Python, Ruby, and Go with support for the web containers Tomcat, Passenger, Puma, and Docker.

AWS Trusted Advisor draws upon best practices learned from the aggregated operational history of serving over a million AWS customers. AWS Trusted Advisor is accessed in the AWS Management Console. Additionally, programmatic access to AWS Trusted Advisor is available with the AWS Support API.

- ✓ AWS Trusted Advisor provides best practices in four categories: cost optimization, security, fault tolerance, and performance improvement
- ✓ **Security Groups–Specific Ports Unrestricted** Checks security groups for rules that allow unrestricted access (0.0.0.0/0) to specific ports

AWS Config

AWS Config is a fully managed service that provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance. With AWS Config, you can discover existing and deleted AWS resources, determine your overall compliance against rules, and dive into configuration details of a resource at any point in time.

- ✓ AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related and how they were configured in the past so that you can see how the configurations and relationships change over time.
- ✓ Use Cases
  - **Discovery**
  - **Change Management**
  - **Continuous Audit and Compliance** AWS Config and AWS Config Rules are designed to help you assess compliance with internal policies and regulatory standards by providing visibility into the configuration of a resource at any time and evaluating relevant configuration changes against rules that you can define.
  - **Troubleshooting**
  - **Security and Incident Analysis**

- ✓ **AWS Config integrates with AWS CloudTrail**, a service that records AWS API calls for an account and delivers API usage log files to an Amazon S3 bucket. If the configuration change of a resource was the result of an API call, AWS Config also records the AWS CloudTrail event ID that corresponds to the API call that changed the resource's configuration. Organizations can then leverage the AWS CloudTrail logs to obtain details of the API call that was made— including who made the API call, at what time, and from which IP address—to use for troubleshooting purposes.
- ✓ Organizations can use the AWS Management Console, API, or AWS CLI to obtain details of **what a resource's configuration looked like at any point in the past**. AWS Config will also automatically deliver a history file to the Amazon S3 bucket you specify every six hours that contains all changes to your resource configurations.

You should architect your AWS usage to take advantage of multiple regions and Availability Zones. Distributing applications across multiple Availability Zones provides the ability to remain resilient in the face of most failure modes, including natural disasters or system failures.

It is not possible for a virtual instance running in promiscuous mode to receive or "sniff" traffic that is intended for a different virtual instance.

Attacks such as Address Resolution Protocol (ARP) cache poisoning do not work within Amazon EC2 and Amazon VPC.

Incident
When you move computer systems and data to the cloud, security responsibilities become shared between you and your cloud service provider.

AWS LAB Assignment

1) VPC (15 marks)
   a. Create one VPC--- 172.20.0.0/16 (10.20.0.0/16)

   b. Create three subnets 172.20.1.0/24 , 172.20.2.0/24, 172.20.3.0/24

SUBNETS resides within ONE Availability Zone and can not span zones.
ONE subnet equals one availability zone

   c. route tables – public – 172.20.1.0/24, private – 172.20.2.0/24, 172.20.3.0/24

https://us-east-2.console.aws.amazon.com/vpc/home?region=us-east-2#routetables:

You should remember the following points about route tables:

1. Your VPC has an implicit router.
2. Your VPC automatically comes with a main route table that you can modify.
3. You can create additional custom route tables for your VPC.
4. Each subnet must be associated with a route table, which controls the routing for the subnet. If you don't explicitly associate a subnet with a particular route table, the subnet uses the main route table.
5. You can replace the main route table with a custom table that you've created so that each new subnet is automatically associated with it.
6. Each route in a table specifies a destination CIDR and a target; for example, traffic destined for 172.16.0.0/12 is targeted for the VPG. AWS uses the most specific route that matches the traffic to determine how to route the traffic

2) IAM (10 Marks)

a. Create one IAM user with read only access to all services; also create access key, secret access key & password.

b. Create another IAM user with EC2 full access.

c. Create IAM Role with S3-full access (attach it to the below instance).

3) EC2 (10 Marks)

a. Launch one instance in public subnet (t2.micro & Amazon linux).

b. Instances is accessible from anywhere through port – 22,80,443,8443,8080

c. All traffic is open within same security group

d. Create 10 gb EBS GP2 volume & attach it to instance (no need to mount).

4) S3 & CloudFront (10 Marks)

a. Create S3 Bucket & put 2 objects in that bucket

b. Make one public and other private

c. Create cloudfront distribution of the same bucket (distribute only public object)

5) CloudWatch & SNS (5 Marks)

a. Create cloud watch event rule for same instance – change in resource state.

b. Push events from cloud watch to SNS Topic

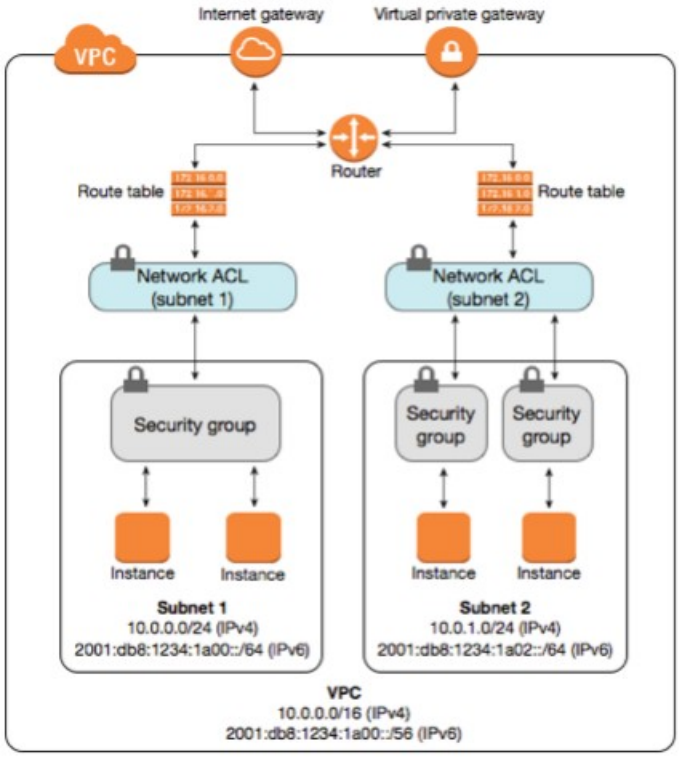
Comparison of Security Groups and Network ACLs

The following table summarizes the basic differences between security groups and network ACLs.

Security Group	Network ACL
Operates at the instance level (first layer of defense)	Operates at the subnet level (second layer of defense)
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules

Security Group	Network ACL
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (backup layer of defense, so you don't have to rely on someone specifying the security group)

The following diagram illustrates the layers of security provided by security groups and network ACLs. For example, traffic from an Internet gateway is routed to the appropriate subnet using the routes in the routing table. The rules of the network ACL associated with the subnet control which traffic is allowed to the subnet. The rules of the security group associated with an instance control which traffic is allowed to the instance.



Default Security Group for Your VPC

Your VPC automatically comes with a default security group. Each EC2 instance that you launch in your VPC is automatically associated with the default security group if you don't specify a different security group when you launch the instance.

The following table describes the default rules for a default security group.

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The security group ID (sg-*xxxxxxxx*)	All	All	Allow inbound traffic from instances assigned to the same security group.
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	All	All	Allow all outbound IPv4 traffic.
::/0	All	All	Allow all outbound IPv6 traffic. This rule is added by default if you create a VPC with an IPv6 CIDR block or if you associate an IPv6 CIDR block with your existing VPC.

Default Network ACL

The default network ACL is configured to allow all traffic to flow in and out of the subnets to which it is associated. Each network ACL also includes a rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied. You can't modify or remove this rule.

The following is an example default network ACL for a VPC that supports IPv4 only.

Inbound					
Rule #	**Type**	**Protocol**	**Port Range**	**Source**	**Allow/Deny**
100	All IPv4 traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 traffic	All	All	0.0.0.0/0	DENY
Outbound					
Rule #	**Type**	**Protocol**	**Port Range**	**Destination**	**Allow/Deny**
100	All IPv4 traffic	all	all	0.0.0.0/0	ALLOW
*	All IPv4 traffic	all	all	0.0.0.0/0	DENY

If you create a VPC with an IPv6 CIDR block or if you associate an IPv6 CIDR block with your existing VPC, we automatically add rules that allow all IPv6 traffic to flow in and out of your subnet. We also add rules whose rule numbers are an asterisk that ensures that a packet is denied if it doesn't match any of the other numbered rules. You can't modify or remove these rules. The following is an example default network ACL for a VPC that supports IPv4 and IPv6.

> **Note**
> If you've modified your default network ACL's inbound rules, we do not automatically add an ALLOW rule for inbound IPv6 traffic when you associate an IPv6 block with your VPC. Similarly, if you've modified the outbound rules, we do not automatically add an ALLOW rule for outbound IPv6 traffic.

Routing Your VPC has an implied router (shown in the configuration diagram above). In this scenario, the VPC wizard creates a custom route table that routes all traffic destined for an address outside the VPC to the Internet gateway, and associates this route table with the subnet. The following table shows the route table for the example in the configuration diagram above. **The first entry is the default entry for local IPv4 routing in the VPC; this entry enables the instances in this VPC to communicate with each other. The second entry routes all other IPv4 subnet traffic to the Internet gateway (for example, igw-1a2b3c4d).**

Security AWS provides two features that you can use to increase security in your VPC: **security groups and network ACLs.**

Security groups control inbound and outbound traffic for your instances, and **network ACLs control inbound and outbound traffic for your subnets**.

In most cases, **security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC.**

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*igw-id*

Routing for IPv6

If you associate an IPv6 CIDR block with your VPC and subnet, your route table must include separate routes for IPv6 traffic. The following table shows the custom route table for this scenario if you choose to enable IPv6 communication in your VPC. The second entry is the default route that's automatically added for local routing in the VPC over IPv6. The fourth entry routes all other IPv6 subnet traffic to the Internet gateway.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	*igw-id*
::/0	*igw-id*

Your VPC comes with a default security group (p. 123). **An instance that's launched into the VPC is automatically associated with the default security group if you don't specify a different security group during launch.**

For this scenario, we recommend that you create the following security groups instead of using the default security group:

• **WebServerSG:** Specify this security group when you launch the web servers in the public subnet.

• DBServerSG: Specify this security group when you launch the database servers in the private subnet.

The instances assigned to a security group can be in different subnets. However, in this scenario, each security group corresponds to the type of role an instance plays, and each role requires the instance to be in a particular subnet.

Therefore, in this scenario, all instances assigned to a security group are in the same subnet.

The following table describes the recommended rules for the WebServerSG security group, which allow the web servers to **receive Internet traffic, as well as SSH and RDP traffic from your network**.

Because the web server doesn't initiate any other outbound communication, the default outbound rule is removed.

The web servers can also initiate read and write requests to the database servers in the private subnet, and send traffic to the Internet; for example, to get software

updates.

WebServerSG: Recommended Rules

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv4 address.
0.0.0.0/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv4 address.
Your home network's public IPv4 address range	TCP	22	Allow inbound SSH access to Linux instances from your home network (over the Internet gateway). You can get the public IPv4 address of your local computer using a service such as http://checkip.amazonaws.com or https://checkip.amazonaws.com. If you are connecting through an ISP or from behind your firewall without a static IP address, you need to find

			out the range of IP addresses used by client computers.
Your home network's public IPv4 address range	TCP	3389	Allow inbound RDP access to Windows instances from your home network (over the Internet gateway).

Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
The ID of your DBServerSG security group	TCP	1433	Allow outbound Microsoft SQL Server access to the database servers assigned to the DBServerSG security group.
The ID of your DBServerSG security group	TCP	3306	Allow outbound MySQL access to the database servers assigned to the DBServerSG security group.
0.0.0.0/0	TCP	80	Allow outbound HTTP access to any IPv4 address.
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to any IPv4 address.

The following table describes the recommended rules for the DBServerSG security group, **which allow read or write database requests from the web servers. The database servers can also initiate traffic bound for the Internet (the route table sends that traffic to the NAT gateway, which then forwards it to the Internet over the Internet gateway**).

DBServerSG: Recommended Rules

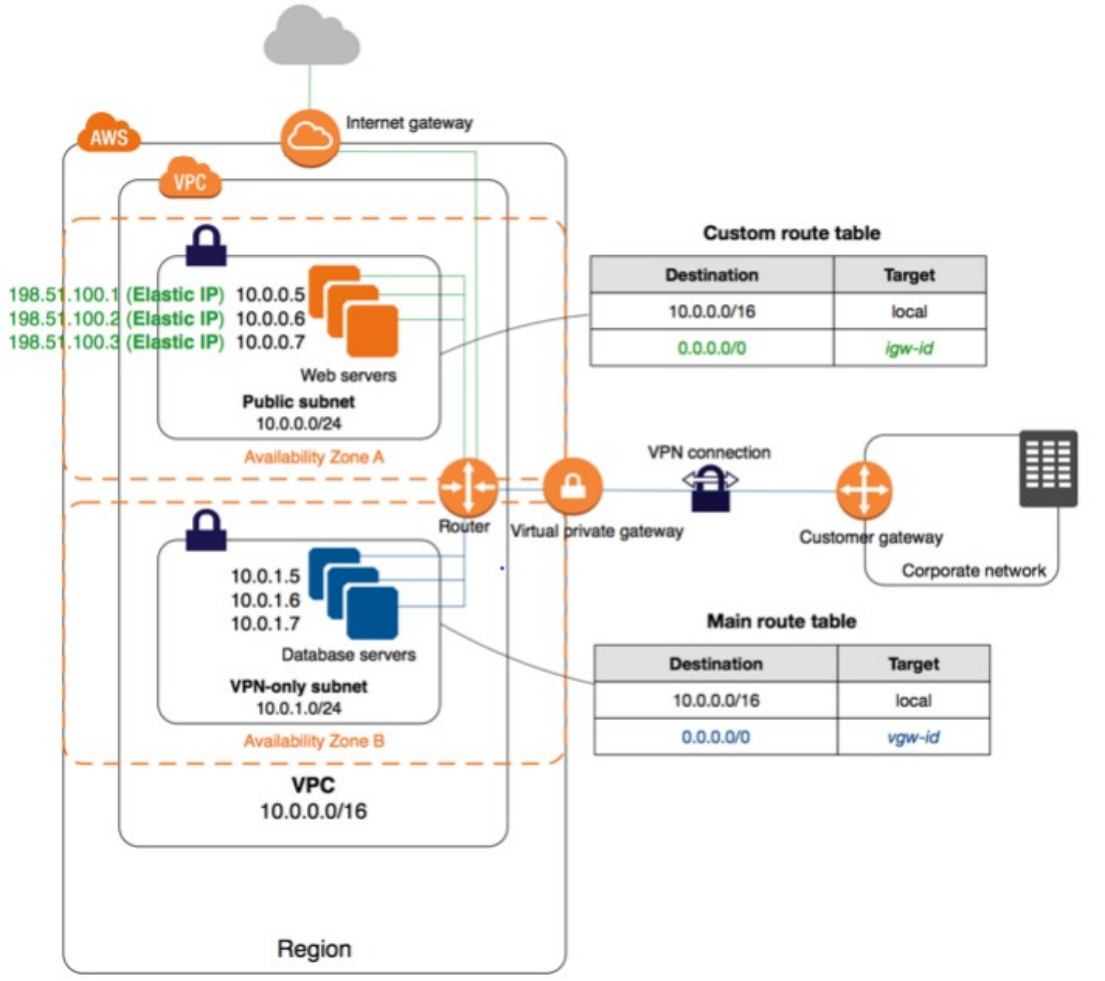
Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The ID of your WebServerSG security group	TCP	1433	Allow inbound Microsoft SQL Server access from the web servers associated with the WebServerSG security group.
The ID of your WebServerSG security group	TCP	3306	Allow inbound MySQL Server access from the web servers associated with the WebServerSG security group.
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet over IPv4 (for example, for software updates).
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet over IPv4 (for example, for software updates).

(Optional) The default security group for a VPC has rules that automatically allow assigned instances to communicate with each other. To allow that type of communication for a custom security group, you must add the following rules:

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The ID of the security group	All	All	Allow inbound traffic from other instances assigned to this security group.
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
The ID of the security group	All	All	Allow outbound traffic to other instances assigned to this security group.

(Optional) If you launch a bastion host in your public subnet to use as a proxy for SSH or RDP traffic from your home network to your private subnet, add a rule to the DBServerSG security group that allows inbound SSH or RDP traffic from the bastion instance or its associated security group.

The following diagram shows the key components of the configuration for this scenario.



Important
For this scenario, the *Amazon VPC Network Administrator Guide* describes what your network administrator needs to do to configure the Amazon VPC customer gateway on your side of the VPN connection.

The configuration for this scenario includes the following:

• A virtual private cloud (VPC) with a size /16 IPv4 CIDR (example: 10.0.0.0/16). This provides 65,536 private IPv4 addresses. 45 Amazon Virtual Private Cloud User Guide Overview

• **A public subnet with a size /24 IPv4 CIDR (example: 10.0.0.0/24). This provides 256 private IPv4 addresses.** A public subnet is a subnet that's associated with a route table that has a route to an Internet gateway.

• A VPN-only subnet with a size /24 IPv4 CIDR (example: 10.0.1.0/24). This provides 256 private IPv4 addresses.

• **An Internet gateway. This connects the VPC to the Internet and to other AWS products.**

• ***A VPN connection between your VPC and your network. The VPN connection consists of a virtual private gateway located on the Amazon side of the VPN connection and a customer gateway located on your side of the VPN connection.***

• Instances with private IPv4 addresses in the subnet range (examples: 10.0.0.5 and 10.0.1.5), which enables the instances to communicate with each other and other instances in the VPC.

• **Instances in the public subnet with Elastic IP addresses (example: 198.51.100.1), which are public IPv4 addresses that enable them to be reached from the Internet. The instances can have public IPv4 addresses assigned at launch instead of Elastic IP addresses.**

Instances in the VPN-only subnet are back-end servers that don't need to accept incoming traffic from the Internet, but can send and receive traffic from your network.

• A custom route table associated with the public subnet. This route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC, and an entry that enables instances in the subnet to communicate directly with the Internet.

• The main route table associated with the VPN-only subnet. The route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC, and an entry that enables instances in the subnet to communicate directly with your network.

Routing

Your VPC has an implied router (shown in the configuration diagram for this scenario). In this scenario, the VPC wizard updates the main route table used with the VPN-only subnet, and creates a custom route table and associates it with the public subnet.

The instances in the VPN-only subnet can't reach the Internet directly; any Internet-bound traffic must first traverse the virtual private gateway to your network, where the traffic is then subject to your firewall and corporate security policies. If the instances send any AWS-bound traffic (for example, requests to the Amazon S3 or Amazon EC2 APIs), the requests must go over the virtual private

gateway to your network and then egress to the Internet before reaching AWS. Currently, we do not support IPv6 for VPN connect

Main Route Table

The first entry is the default entry for local routing in the VPC; this entry enables the instances in the VPC to communicate with each other over IPv4. The second entry routes all other IPv4 subnet traffic from the private subnet to your network over the virtual private gateway (for example, `vgw-1a2b3c4d`).

Destination	Target
`10.0.0.0/16`	local
`0.0.0.0/0`	*vgw-id*

Custom Route Table

The first entry is the default entry for local routing in the VPC; this entry enables the instances in the VPC to communicate with each other. The second entry routes all other IPv4 subnet traffic from the public subnet to the Internet over the Internet gateway (for example, `igw-1a2b3c4d`).

Destination	Target
`10.0.0.0/16`	local
`0.0.0.0/0`	*igw-id*

Rules for the WebServerSG Security Group The following table describes the inbound and outbound rules for the WebServerSG security group. You'll add the inbound rules yourself. The outbound rule is a default rule that allows all outbound communication to anywhere — you do not need to add this rule yourself.

Inbound			
Source IP	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	TCP	80	Allows inbound HTTP access from any IPv4 address.
0.0.0.0/0	TCP	443	Allows inbound HTTPS access from any IPv4 address.
Public IPv4 address range of your home network	TCP	22	Allows inbound SSH access from your home network to a Linux/UNIX instance.
Public IPv4 address range of your home network	TCP	3389	Allows inbound RDP access from your home network to a Windows instance.
Outbound			
Destination IP	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	All	All	The default outbound rule that allows all outbound IPv4 communication.

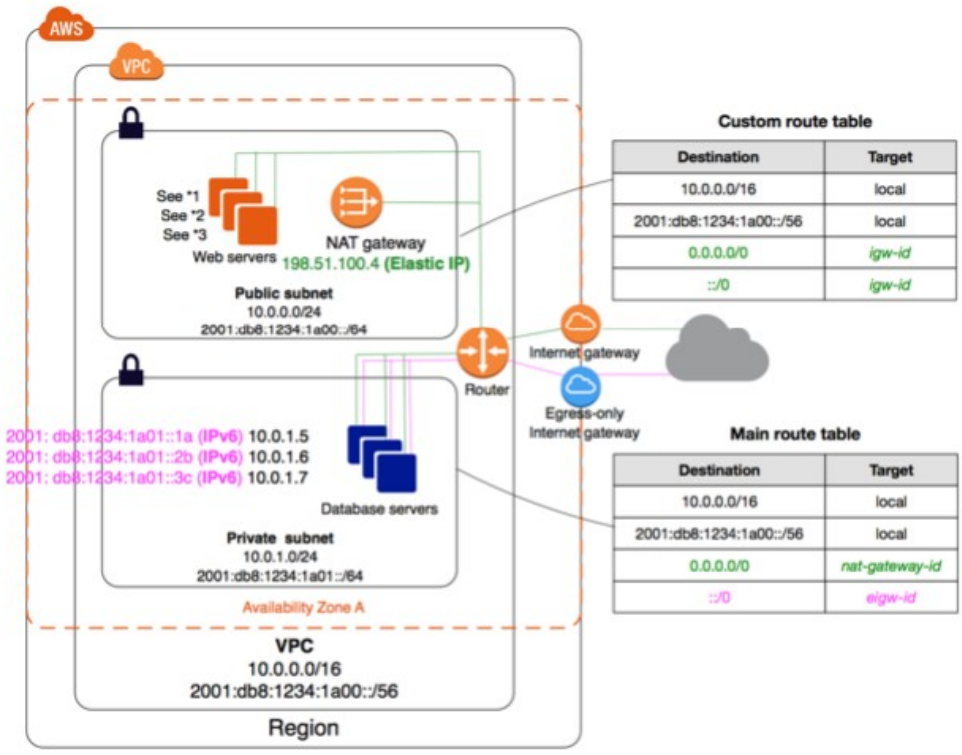
Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The security group ID (sg-*xxxxxxxx*)	All	All	Allow inbound traffic from instances assigned to the same security group.
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	All	All	Allow all outbound IPv4 traffic.
::/0	All	All	Allow all outbound IPv6 traffic. This rule is added by default if you create a VPC with an IPv6 CIDR block or if you associate an IPv6 CIDR block with your existing VPC.

source security group to access instances in the security group. This does not add rules from the source security group to this security group. Incoming traffic is allowed based

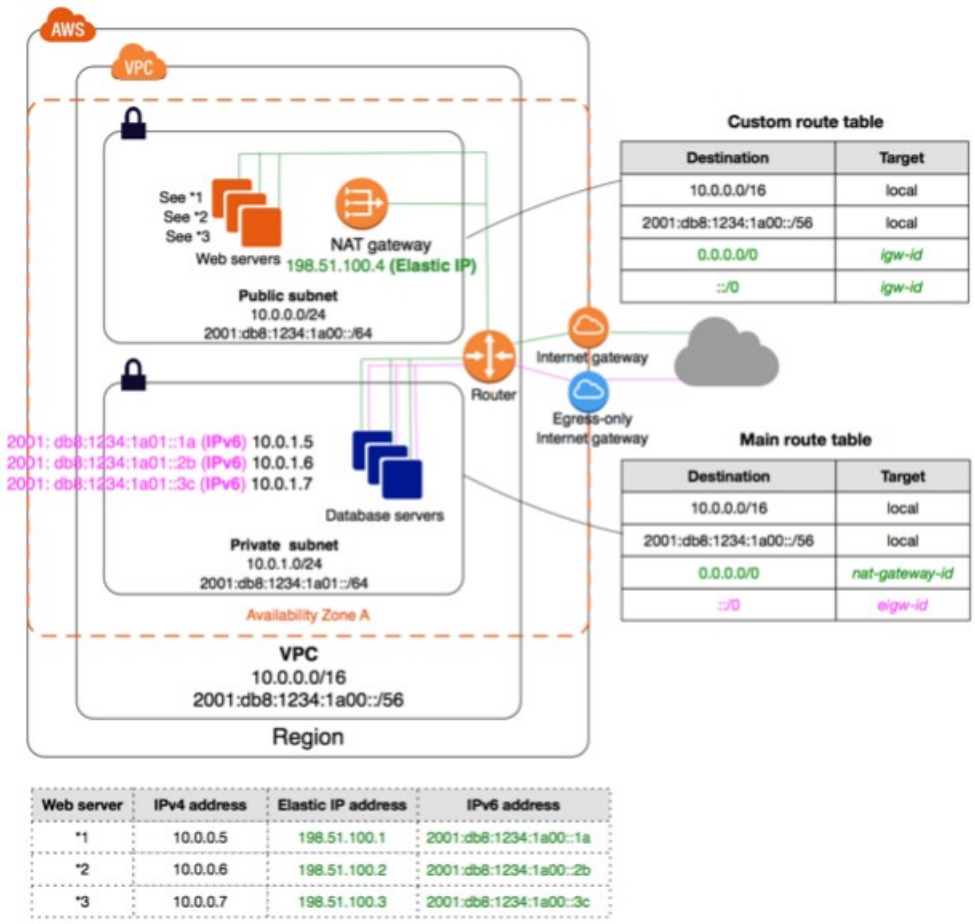
on the private IP addresses of the instances that are associated with the source security group (and not the public IP or Elastic IP

The kind of rules you add may depend on the purpose of the instance.
The following table describes example rules for a security group for web servers. The web servers can receive HTTP and HTTPS traffic
from all IPv4 and IPv6 addresses, and send SQL or MySQL traffic to a database server.

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	TCP	80	Allow inbound HTTP access from all IPv4 addresses
::/0	TCP	80	Allow inbound HTTP access from all IPv6 addresses
0.0.0.0/0	TCP	443	Allow inbound HTTPS access from all IPv4 addresses
::/0	TCP	443	Allow inbound HTTPS access from all IPv6 addresses
Your network's public IPv4 address range	TCP	22	Allow inbound SSH access to Linux instances from IPv4 IP addresses in your network (over the Internet gateway)
Your network's public IPv4 address range	TCP	3389	Allow inbound RDP access to Windows instances from IPv4 IP addresses in your network (over the Internet gateway)
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
The ID of the security group for your database servers	TCP	1433	Allow outbound Microsoft SQL Server access to instances in the specified security group
The ID of the security group for your MySQL database servers	TCP	3306	Allow outbound MySQL access to instances in the specified security group



Web server	IPv4 address	Elastic IP address	IPv6 address
*1	10.0.0.5	198.51.100.1	2001:db8:1234:1a00::1a
*2	10.0.0.6	198.51.100.2	2001:db8:1234:1a00::2b
*3	10.0.0.7	198.51.100.3	2001:db8:1234:1a00::3c



Web server	IPv4 address	Elastic IP address	IPv6 address
*1	10.0.0.5	198.51.100.1	2001:db8:1234:1a00::1a
*2	10.0.0.6	198.51.100.2	2001:db8:1234:1a00::2b
*3	10.0.0.7	198.51.100.3	2001:db8:1234:1a00::3c

AWS Security Essentials

- Region
- Availability Zones
- Endpoints
- IAM
- Compliance

VPC Endpoint :

- **Allows for a private connection to AWS service without going through the internet.**
- Traffic does not leave the VPC network

- VPC Endpoints are virtual devices and have scalable, redundant and highly available.

2 types of VPC Endpoints

Interface (using AWS Private Link):

- An Elastic Network Interface (ENI) with a private address server as the endpoint
  - Kinesis streams
  - Elastic load Balancing
  - EC2 API
  - EC2 System Manger
  - Service Catalog
- Gateway:
  - A target for a route table in your environment
    Supported Services
  - DynamoDB
  - S3

LIMIATIONS

- Same region only
- IPv4 only
- An interface endpoint cannot be accessed through a VPN or VPC peering connection.
  ONLY DIRECT CONNECT

IAM and Compliance

- Identity and Access Management

- Global Scope across all of AWS (all regions)

- Allows for large-scale granulatiry

Allows for central management:

- Users
- Password
- Access Keys
- Permissions
- Groups
- Roles

IAM is one of the main topics in AWS Security.

AWS Cloud Compliance

Documents that AWS meets regulatory, audit, and security standards

- HIPAA
- ISO Standards
- Various regulatory and security agencies around the world.

ONLY APPLIES TO THE SERVICE AND INFRASTURCE THAT AWS IS RESPONSIBLE FOR.

Does not mean that the application and data that you deploy in your AWS environment are compliant.

Shared Responsibility Model and Trusted Advisor

The Shared Responsibility Model describes what Amazon Web Service is responsible for and what YOU, the USER or CUSTOMER, is responsible for when it related to SECURITY.

AWS Infrastructure Service Includes like **Amazon Virtual Private Cloud (VPC)** , Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS) Auto Scaling	
Amazon is responsible for	Customer is responsible for
Regions, AZs, Edge Location	Customer Data
Compute, Storage, Database, Networking	Platforms and Applications
	OS and Network Configuration
	➢ Patching, ➢ Security Groups, ➢ Network Access Control
	IAM for Customer ➢ Password ➢ Access Keys, ➢ Permissions
	Additional Concerns: Data in Motion, Data AT Rest, Data In Use **Data Encryption** **Data Integrity**

AWS Container Service *Service like Amazon Relational Database Service (Amazon RDS), Amazon Elastic*

MapReduce (Amazon EMR), Amazon EC2 Container Service (Amazon ECS),	
Amazon is responsible for	Customer is responsible for
Regions, AZs, Edge Location	Customer Data
Compute, Storage, Database, Networking	Data Integrity
Platforms and Applications	Additional Concerns: **Data Encryption** **Data Integrity**
OS and network configuration	

AWS Abstracted Services	
Amazon is responsible for	Customer is responsible for
Regions, AZs, Edge Location	Customer IAM
Compute, Storage, Database, Networking	Data in transit and Client-side
Platforms and Applications	
OS and network configuration	
Network traffic protection	

AWS Trusted Advisor tool and what it can show you about your resources and environments.
Allows an AWS customers to get report on their environment

- ✓ Cost Optimization
- ✓ Performance
- ✓ Security
- ✓ Fault Tolerance

Available to all customers

Access to six core checks Security

1. Security Groups
2. Identity and Access Management
3. Multi Factor Authentication on ROOT account
4. EBS
5. RDS public snapshots
6. Performance (services limits)

Available to Business and Enterprise support plans:

Access to the full set of checks

- ✓ Cost Optimization
- ✓ Performance
- ✓ Security
- ✓ Fault Tolerance

Notification

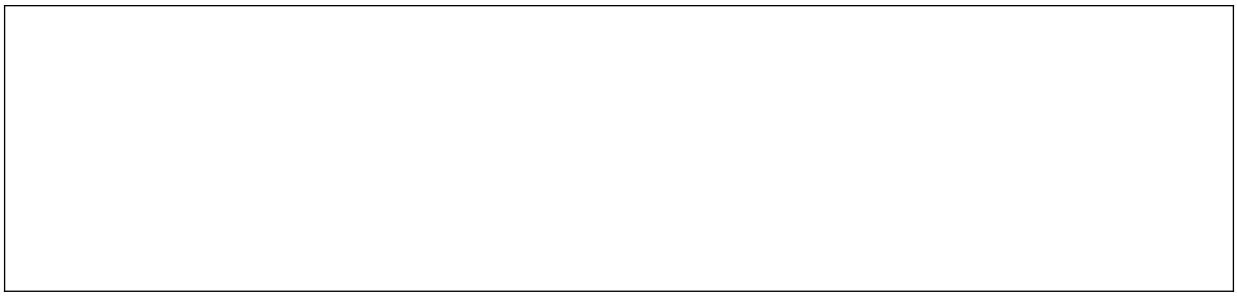
- ✓ Weekly updates

Programmatic Access

- ✓ Retrieve results from the AWS support API

Identify and Access Management

1. Delete your root access keys
2. Activate Multi Factor Authentication on your root account
3. Create individual IAM users
4. Use groups to assign permissions
5. Apply an IAM password policy.



Infrastructure Security

- Use SSL/TLS endpoints for your applications
- Security Groups and network access control list (NACLs)
- Routing and placement must be planned (Public and Private subnets)
- Build your own VPN Solutions

IPSec Tunnel over Internet

- Customer Gateway, VPN Gateway, VPN Connection (AWS VPN comonents)
- Custom VPN solutions if required
- VPC networking (subnets, security group and network access control list)

AWS Direct Connect

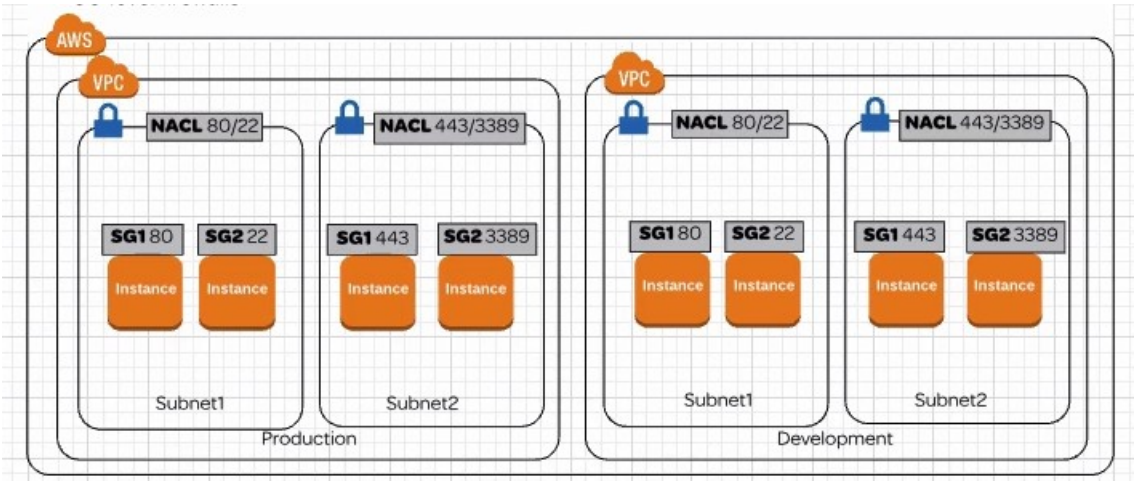
- Using private peered connection which might not need additional security
- VPC networking (subnets, security group and network access control list)

Network Segmentations

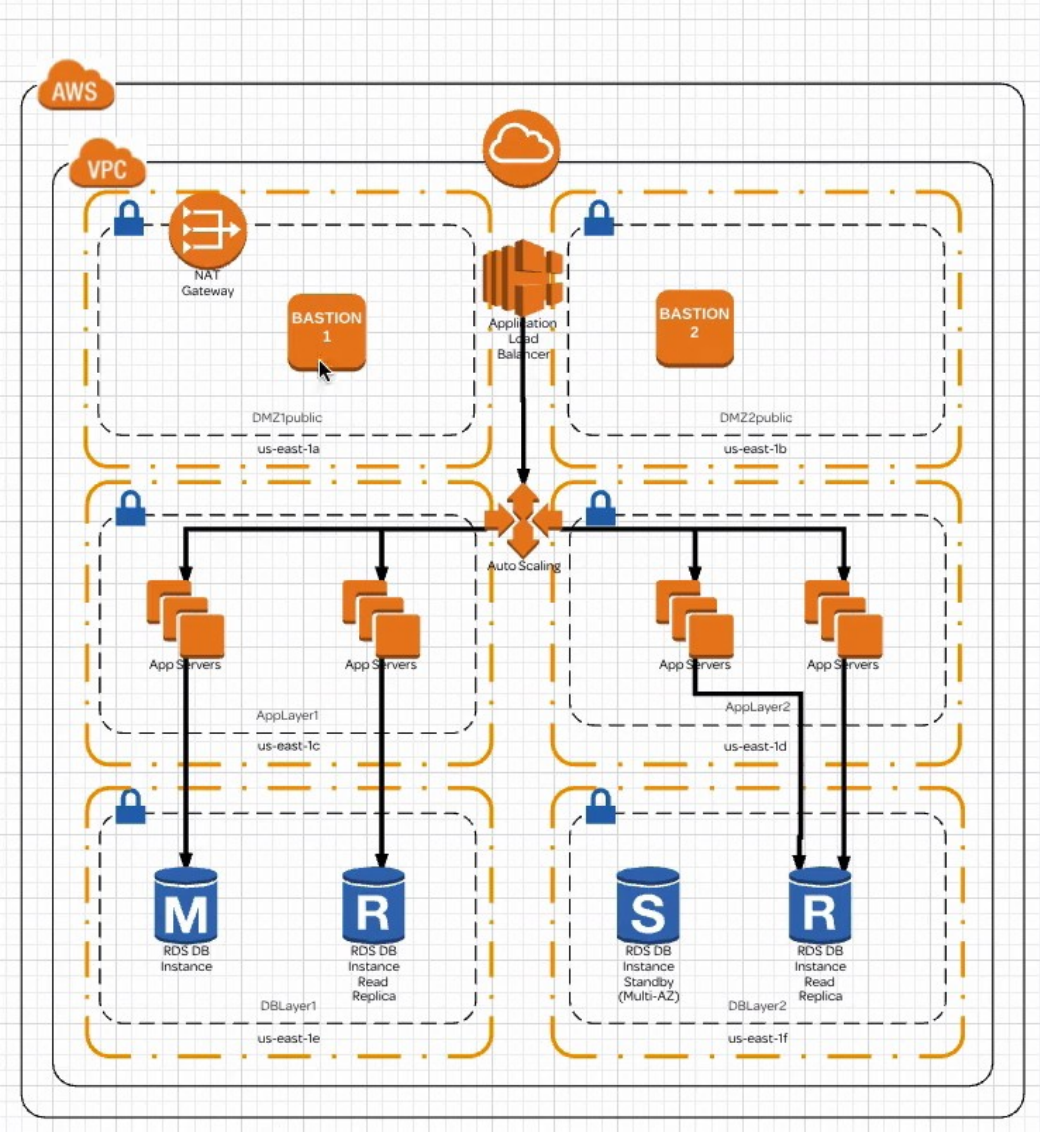
- VPC (Virtual Private Cloud)
  - Isolate workloads into separate VPCs (based on application, department, Production, Development, and Test and DR)
- Security Group
  - Group instances (Guest VM) with similar functions
  - Stateul = every allowed TCP or UDP port will be allowed in both directions
- Network access control list
  - Stateless = inbound and outbound rules are separate, no dependencies

- Granular control over IP protocols
- Work with security group
- Ephemeral ports – Client request depending on OS (1024- 65535)

- Host-based firewall
  - OS-level firewalls



Three tier application (Web in DMZ, Apps in the middle and DB in the last



Inbound Rule Security Group Automatic Load Balancing				
Type	Protocol	Port Range	Source	Description
HTTP	Tcp	80	0.0.0.0/0	
HTTPS	tcp	443	0.0.0.0/0	
Inbound Bastion Host				
Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	

Security Group for App Servers				
Type	Protocol	Port Range	Source	Description
Http	TCP	80	Sg-882e1aft	ALB
Ssh	TCp	22	SG-Basion	Basion Security Group
RDS DB				
Type	Protocol	Port Range	Source	Description
MySQL/Aurora	TCP	3306	Sg-Apps	App Servers can only talk to the databases

Inbound Network Access Control List for DMZ					
Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH 22	TCP	22	0.0.0.0/0	Allow
110	HTTP	TCP	80	0.0.0.0/0	Allow
120	HTTPS	TCp	443	0.0.0.0/0	Allow
130	Cutom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow
Outbound Network Access Control List for DMZ					
Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH 22	TCP	22	0.0.0.0/0	Allow
110	HTTP	TCP	80	0.0.0.0/0	Allow
120	HTTPS	TCP	443	0.0.0.0/0	Allow
130	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow

From DMZ traffic to App Server for network access control List

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH 22	TCP	22	0.0.0.0/0	Allow
110	HTTP	TCP	80	0.0.0.0/0	Allow
120	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow
*	All Traffic	ALL	ALL	0.0.0.0/0	DENY

Inbound Traffic for DMZ traffic to App Server for network access control List

	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH 22	TCP	22	0.0.0.0/0	Allow
110	HTTP	TCP	80	0.0.0.0/0	Allow
120	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow
*	All Traffic	ALL	ALL	0.0.0.0/0	DENY

INBOUND From DMZ traffic to App Server for network access control List

Rule	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH 22	TCP	22	0.0.0.0/0	Allow
110	HTTP	TCP	80	0.0.0.0/0	Allow
120	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow
*	All Traffic	ALL	ALL	0.0.0.0/0	DENY
	OUTBOUND From DMZ traffic to App Server for network access control List				

Rule	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH 22	TCP	22	0.0.0.0/0	Allow
110	HTTP	TCP	80	0.0.0.0/0	Allow
120	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow
	All Traffic	ALL	ALL	0.0.0.0/0	DENY

Inbound for Database Subnet					
Rule number	Type	Protocol	Port Range	Source	Allow / Deny
100	MySOL		3306	0.0.0.0/0	Allow
110	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow
Outbound Rules					
100	MySOL	TCP	3306		
110	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	Allow

Access Advisor -

Strengthening and Threat Protection Layer

Encryption

Server-side Encryption

- Data is encrypted as it is written to disk then decrypted as it is read from the disk
- Often referred to as encryption "AT REST"

Client-side Encryption

- Data is encrypted by the client before it is sent to the server, then decrypted when the client receives data from the server.
- Often referred to as encryption "IN TRANSIT".

Symmetric Encryption

- Uses the same key to encrypt and decrypt
- Example Advance Encryption Standard (AES) 128, 192, and 256 bit

Asymmetric Encryption

- Uses different keys to encrypt and decrypt, a public and a private key
- The public key is available to any entity
- Example: Secure Sockets Layer (SSL), Transport Layer Security (TLS), SSH.

HSM and KMS

Hardware Security Module
Physical device use to security storage and key management

Key Management System

Managed service that allows you to create and control your encryption keys

- Advantage over HSM are:
- Can use IAM policies for KMS access
- AWS services integrate directly with KMS

Important Concepts

- KMS store Customer Master Keys (CMK)
- The process follows symmetric encryption but has a major twist.
- In some HSMs, there can be any number of key encryption keys (KEK)
- Process known as enveloping
- KMS only envelops one layer and store the “TOP” key, the CMK
-
-

Data protection fundamentals and regulatory constraints.

Data in the cloud should be perceived to have the same needs and properties as data in the legacy enviorment

Where data being stored?

How is data recovered when needed?

How is it ported to the archive on regular basis?

Where encryption keys are store can affect the over all risk of the data in severe ways.

What is AWS CloudFormation?

AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; AWS CloudFormation handles all of that. The following scenarios demonstrate how AWS CloudFormation can help.

AWS CloudFormation Concepts

When you use AWS CloudFormation, **you work with *templates* and *stacks***. You create templates to describe your AWS resources and their properties. Whenever you create a stack, AWS CloudFormation provisions the resources that are described in your template.

Topics

- Templates
- Stacks
- Change Sets

Templates

An AWS CloudFormation template is a JSON or YAML formatted text file. You can save these files with any extension, such as .json, .yaml, .template, or .txt. AWS CloudFormation uses these templates as blueprints for building your AWS resources. For example, in a template, you can describe an Amazon EC2 instance, such as the instance type, the AMI ID, block device mappings, and its Amazon EC2 key pair name. Whenever you create a stack, you also specify a template that AWS CloudFormation uses to create whatever you described in the template.

For example, if you created a stack with the following template, AWS CloudFormation provisions an instance with an ami-2f726546 AMI ID, t1.micro instance type, testkey key pair name, and an Amazon EBS volume.

For example, if you created a stack with the following template, AWS CloudFormation provisions an instance with an ami-2f726546 AMI ID, t1.micro instance type, testkey key pair name, and an Amazon EBS volume.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Resources:
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-2f726546"
      InstanceType: t1.micro
      KeyName: testkey
      BlockDeviceMappings:
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
  MyEIP:
    Type: AWS::EC2::EIP
    Properties:
      InstanceId: !Ref MyEC2Instance
```

Description:

This template deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an Internet Gateway, with a default route on the public subnets. It deploys a pair of NAT Gateways (one in each AZ),

and default routes for them in the private subnets.

Parameters:

EnvironmentName:

Description: An environment name that will be prefixed to resource names

Type: String

VpcCIDR:

Description: Please enter the IP range (CIDR notation) for this VPC

Type: String

Default: 172.20.0.0/16

PublicSubnet1CIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone

Type: String

Default: 172.20.1.0/24

PublicSubnet2CIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone

Type: String

Default: 172.20.4.0/24

PrivateSubnet1CIDR:

Description: Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone

Type: String

Default: 172.20.2.0/24

PrivateSubnet2CIDR:

Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone

Type: String

Default: 172.20.3.0/24

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Ref EnvironmentName

InternetGateway:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

```
      - Key: Name
        Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AavailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
```

```
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet2CIDR
```

```
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1

NatGateway2:
  Type: AWS::EC2::NatGateway
```

```
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes

DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1
```

```
PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
```

```
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

NoIngressSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "no-ingress-sg"
    GroupDescription: "Security group with no ingress rule"
    VpcId: !Ref VPC
```

Outputs:

VPC:

Description: A reference to the created VPC

Value: !Ref VPC

PublicSubnets:

Description: A list of the public subnets

Value: !Join [",", [!Ref PublicSubnet1, !Ref PublicSubnet2]]

PrivateSubnets:

Description: A list of the private subnets

Value: !Join [",", [!Ref PrivateSubnet1, !Ref PrivateSubnet2]]

PublicSubnet1:

Description: A reference to the public subnet in the 1st Availability Zone

Value: !Ref PublicSubnet1

PublicSubnet2:

Description: A reference to the public subnet in the 1st Availability Zone

Value: !Ref PublicSubnet2

PrivateSubnet1:

Description: A reference to the private subnet in the 1st Availability Zone

Value: !Ref PrivateSubnet1

PrivateSubnet2:

Description: A reference to the private subnet in the 2nd Availability Zone

Value: !Ref PrivateSubnet2

NoIngressSecurityGroup:

Description: Security group with no ingress rule

Value: !Ref NoIngressSecurityGroup

Reference and AWS Resource Types documentation on AWS::EC2::Instance. The reference documentation is going to be your best friend once you get the hang of CloudFormation.

This EC2Instance resource demonstrates a couple of uses of Ref. **Ref is a way to reference values from other parts of the template. For example, Ref: InstanceSecurityGroup refers to the only other resource in this template, the SecurityGroup to be created**. Here's the definition of that resource:

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType:
      Ref: InstanceType
```

```
SecurityGroups:
- Ref: InstanceSecurityGroup
KeyName:
  Ref: KeyName
ImageId:
  Fn::FindInMap:
  - AWSRegionArch2AMI
  - Ref: AWS::Region
  - Fn::FindInMap:
    - AWSInstanceType2Arch
    - Ref: InstanceType
    - Arch
```

Template Sections

Templates include several major sections**. The Resources section is the only required section. Some sections in a template can be in any order. However, as you build your template, it might be helpful to use the logical ordering of the following list, as values in one section might refer to values from a previous section.** The list gives a brief overview of each section.

Format Version (optional)

The AWS CloudFormation template version that the template conforms to. The template format version is not the same as the API or WSDL version. The template format version can change independently of the API and WSDL versions.

Description (optional)

A text string that describes the template. This section must always follow the template format version section.

Metadata (optional)

Objects that provide additional information about the template.

Parameters (optional)

Values to pass to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.

Mappings (optional)

A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. **You can match a key to a corresponding value by using the Fn::FindInMap intrinsic** function in the Resources and Outputs section.

Conditions (optional)

Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

Transform (optional)

For serverless applications (also referred to as Lambda-based applications), specifies the version of the AWS Serverless Application Model (AWS SAM) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it is processed.

You can also use AWS::Include transforms to work with template snippets that are stored separately from the main AWS CloudFormation template. You can store your snippet files in an Amazon S3 bucket and then reuse the functions across multiple templates.

Resources (required)

Specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.

Outputs (optional)

Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name and then call the aws cloudformation describe-stacks AWS CLI command to view the name.

The following example shows a YAML-formatted template fragment.

AWSTemplateFormatVersion: "*version date*"
Description:
  String
Metadata:
  template metadata
Parameters:
  set of parameters
Mappings:
  set of mappings
Conditions:
  set of conditions
Transform:
  set of transforms
Resources:
  set of resources

Outputs:
  set of outputs

Format Version

The **AWSTemplateFormatVersion** section (optional) identifies the capabilities of the template. The latest template format version is **2010-09-09** and is currently the only valid value.

Parameters

Use the optional Parameters section to customize your templates. **Parameters enable you to input custom values to your template each time** you create or update a stack.

Parameters:

```
  InstanceTypeParameter:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
    Description: Enter m1.small, or m1.large. Default is t2.micro.
```