Kennedy College of Sciences

Computer Science Department

Comp 4600 – 208

Cloud Computing - Spring 24

Prof. Johannes Weis

Handout 4 – Security

AWS In Action Chapter 6

AWS account:

https://620339869704.signin.aws.amazon.com/console

Account Usage policy:

- You all currently have Administrator privileges – a lot of power!
- Use free tier as outlined in class and in the book
- Delete things you create before logging out
-

Not following this policy will yield in loosing account privileges and potentially failing the class

What is Cloud Security?

Cloud security, also known as cloud computing security, is **a collection of security measures designed to protect cloud-based infrastructure, applications, and data**.

Security building blocks

1 ***Installing software updates***—New security vulnerabilities are found in software every day. Software vendors release updates to fix those vulnerabilities and it's your job to install those updates as quickly as possible after they're released. Otherwise your system will be an easy victim for hackers.

2 ***Restricting access to your AWS account***—This becomes even more important if you aren't the only one accessing your AWS account (if coworkers and scripts are also accessing it). A buggy script could easily terminate all your EC2 instances instead of only the one you intended. Granting only the permissions you need is key to securing your AWS resources from accidental or intended disastrous actions.

3 ***Controlling network traffic to and from your EC2 instances***—You only want ports to be accessible if they must be. If you run a web server, the only ports you need to open to the outside world are port 80 for HTTP traffic and 443 for HTTPS traffic. Close down all the other ports!

4 ***Creating a private network in AWS***—You can create subnets that aren't reachable from the internet. And if they're not reachable, nobody can access them. Really, nobody? You'll learn

5 ***Securing your applications*** **–** Many aspects! How to handle and store confidential information? For example, you need to check user input and allow only the necessary characters, don't save passwords in plain text, and use TLS/SSL to encrypt traffic between your virtual machines and your users. If you are installing applications with a package manager for your operating system, using Amazon Inspector (https://aws.amazon.com/inspector/) allows you to run automated security assessments.

Who’s responsible for security?

The cloud is a shared-responsibility environment, meaning responsibility is shared between you and AWS.

AWS is responsible for the following:

- Protecting the network through automated monitoring systems and robust internet access, to prevent Distributed Denial of Service (DDoS) attacks.
- Performing background checks on employees who have access to sensitive areas.
- Decommissioning storage devices by physically destroying them after end of life.
- Ensuring the physical and environmental security of data centers, including fire protection and security staff.

The security standards are reviewed by third parties; you can find an up-to-date overview at http://aws.amazon.com/compliance/.

What are your responsibilities?

- Implementing access management that restricts access to AWS resources like S3 and EC2 to a minimum, using AWS IAM.
- Encrypting network traffic to prevent attackers from reading or manipulating data (for example, using HTTPS).
- Configuring a firewall for your virtual network that controls incoming and outgoing traffic with security groups and ACLs.
- Encrypting data at rest. For example, enable data encryption for your database or other storage systems.
- Managing patches for the OS and additional software on virtual machines.

Security involves an interaction between AWS and you, the customer. If you play by the rules, you can achieve high security standards in the

Cloudformation – install updates on startup (Amazon Linux version)

```
Instance:
Type: 'AWS::EC2::Instance'
Properties:
# [...]
UserData:
'Fn::Base64': |
#!/bin/bash -x
yum -y update
```

To install only security updates, do the following:

```
Instance:
Type: 'AWS::EC2::Instance'
Properties:
# [...]
UserData:
'Fn::Base64': |
#!/bin/bash -x
yum -y --security update
```

To install only specific updates, do the following:

```
yum update-to bash-4.2.46-28.37.amzn1
```

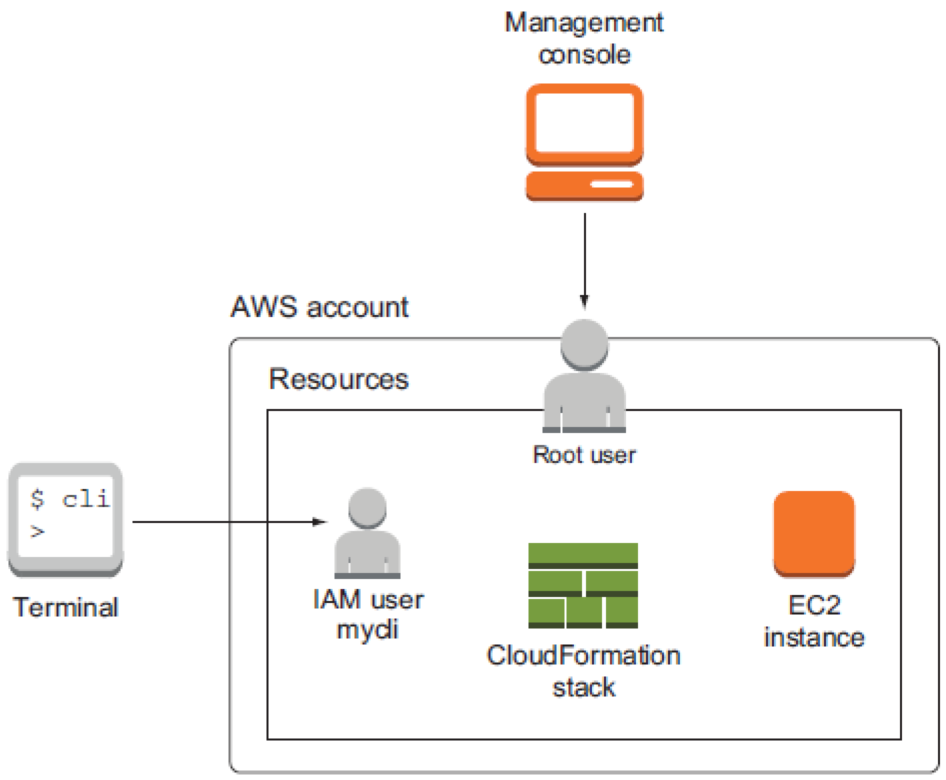
Script to install security updates on all running VM's:

```
PUBLICIPADDRESSES="$(aws ec2 describe-instances \
➥ --filters "Name=instance-state-name,Values=running" \
➥ --query "Reservations[].Instances[].PublicIpAddress" \
➥ --output text)"
for PUBLICIPADDRESS in $PUBLICIPADDRESSES; do
ssh -t "ec2-user@$PUBLICIPADDRESS" \
➥ "sudo yum -y --security update"
Done
```

Other ideas ?

- Recycle systems (for kernel updates)
- Run scheduled jobs (cron) for regular updates

AWS account root user



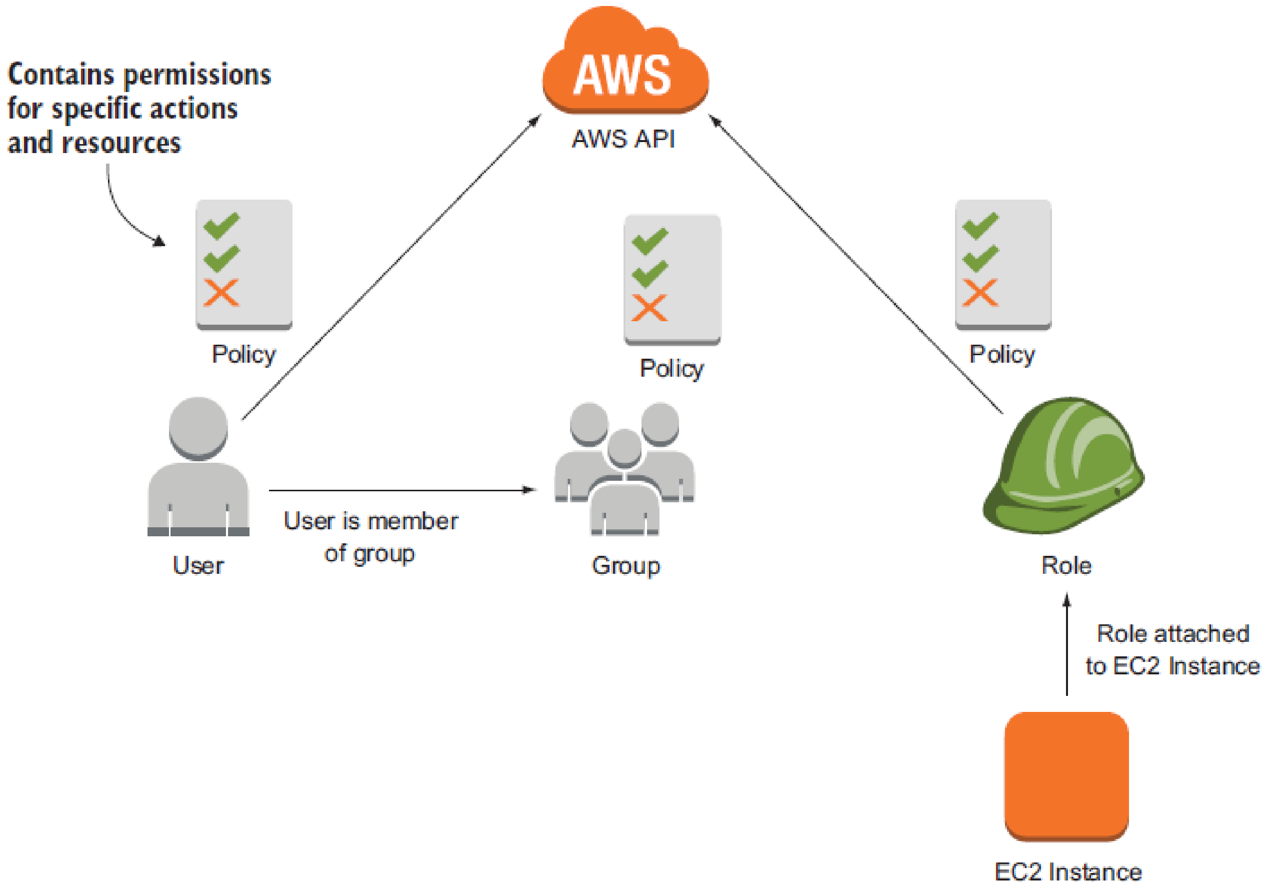
AWS account root user has access to everything

Recommendation: enable MFA for root user / all users – especially if you store confidential data

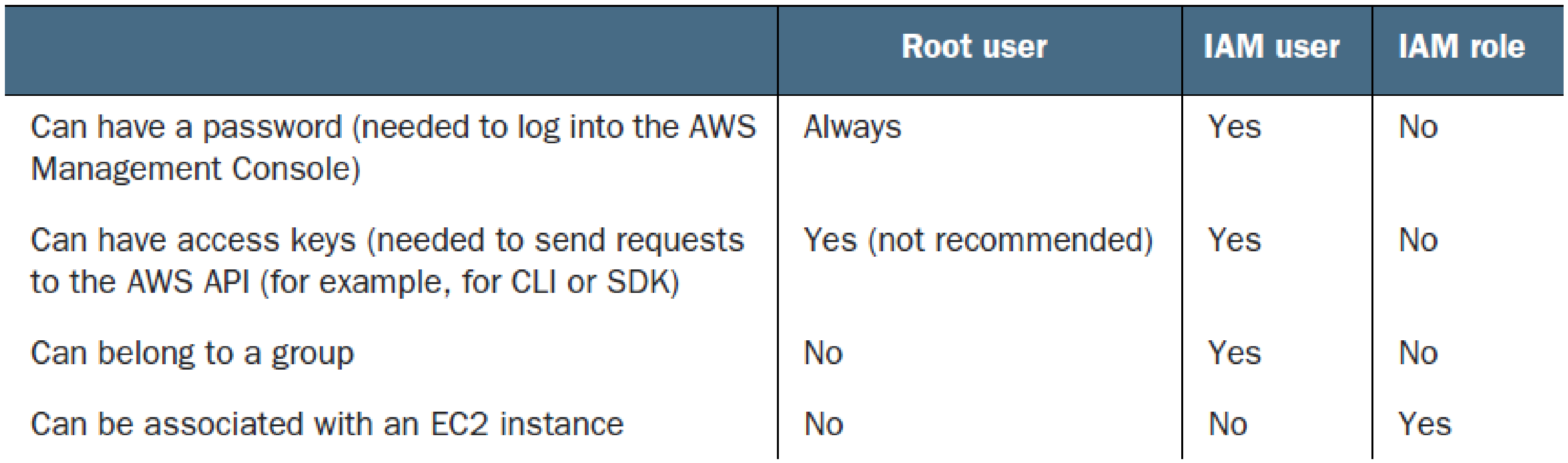
IAM (Identity and access management)

- An **IAM user** is used to authenticate people accessing your AWS account.
- An **IAM group** is a collection of IAM users.
- An **IAM role** is used to authenticate AWS resources, for example an EC2
- instance.
- An **IAM policy** is used to define the permissions for a user, group, or role.

IAM Users, groups, policies and roles


Contains permissions for specific actions and resources
AWS
AWS API
Policy
Policy
Policy
User
User is member of group
Group
Role
Role attached to EC2 Instance
EC2 Instance

Differences between IAM users and roles:



	Root user	IAM user	IAM role
Can have a password (needed to log into the AWS Management Console)	Always	Yes	No
Can have access keys (needed to send requests to the AWS API (for example, for CLI or SDK)	Yes (not recommended)	Yes	No
Can belong to a group	No	Yes	No
Can be associated with an EC2 instance	No	No	Yes

Sample policy – allow all EC2 actions except terminate:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "1",
        "Effect": "Allow",
        "Action": "ec2:*",
        "Resource": "*"
    }, {
        "Sid": "2",
        "Effect": "Deny",
        "Action": "ec2:TerminateInstances",
        "Resource": "*"
    }]
}
```

Get account id from console (IAM dashboard) or cli:

```
aws iam get-user --query "User.Arn" --output text
```

Can limit access to specific instance

```
"Resource":
    ➥ "arn:aws:ec2:us-east-1:111111111111:instance/i-0b5c991e026104db9"
```

Different types of policies:

Managed policy—If you want to create policies that can be reused in your account, a managed policy is what you're looking for. There are two types of managed policies:

- ***AWS managed policy***—A policy maintained by AWS. There are policies that grant admin rights, read-only rights, and so on.
- ***Customer managed***—A policy maintained by you. It could be a policy that represents the roles in your organization, for example.

Inline policy—A policy that belongs to a certain IAM role, user, or group. An inline policy can't exist without the IAM role, user, or group that it belongs to.

Using Inline policies in Cloudformation is convenient

IAM users:

Script:

```
$student="testuser4@student.uml.edu"

aws iam create-user --user-name $student
aws iam add-user-to-group  --group-name UML_students --user-name $student
aws iam create-login-profile  --password resetthispw --password-reset-required --user-name $student
```

Authenticating AWS resources with roles

There are various use cases where an EC2 instance needs to access or manage AWS resources. For example, an EC2 instance might need to :

- Back up data to the object store S3.
- Terminate itself after a job has been completed.
- Change the configuration of the private network environment in the cloud.

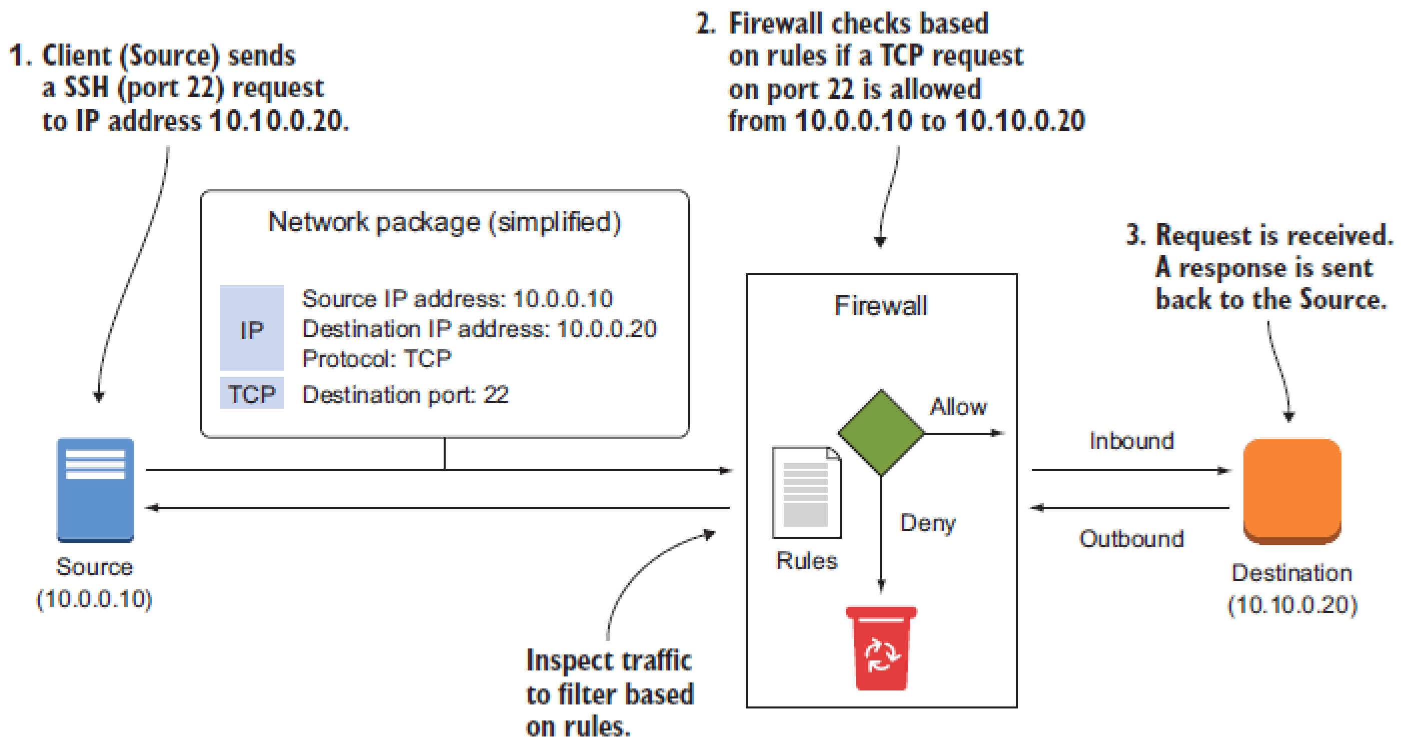
To be able to access the AWS API, an EC2 instance needs to authenticate itself. You could create an IAM user with access keys and store the access keys on an EC2 instance for authentication. But doing so is a hassle, especially if you want to rotate the access keys regularly.

Instead of using an IAM user for authentication, you should use an IAM role whenever you need to authenticate AWS resources like EC2 instances. When using an IAM role, your access keys are injected into your EC2 instance automatically.

Example – allow EC2 to stop itself – see sample code chapter 6 ec2-iamrole.yaml

Controlling Network traffic to your EC2 instances

SSH via Firewall


1. Client (Source) sends a SSH (port 22) request to IP address 10.10.0.20.
2. Firewall checks based on rules if a TCP request on port 22 is allowed from 10.0.0.10 to 10.10.0.20
3. Request is received. A response is sent back to the Source.
Network package (simplified)
IP
Source IP address: 10.0.0.10
Destination IP address: 10.0.0.20
Protocol: TCP
TCP
Destination port: 22
Firewall
Allow
Deny
Rules
Inbound
Outbound
Source (10.0.0.10)
Destination (10.10.0.20)
Inspect traffic to filter based on rules.

Controlling traffic to virtual machines with security groups

Associate a security group with AWS resources such as EC2 instances to control traffic. It's common for EC2 instances to have more than one security group associated with them, and for the same security group to be associated with multiple EC2 instances. A security group consists of a set of rules. Each rule allows network traffic based on the following:

- Direction (inbound or outbound)
- IP protocol (TCP, UDP, ICMP)
- Port
- Source/destination based on IP address, IP address range, or security group

See /chapter06/firewall1.yaml

Allowing ICMP traffic

Internet Control Message Protocol (ICMP) - Needed for ping for example

```
SecurityGroup:
     Type: 'AWS::EC2::SecurityGroup'
     Properties:
          GroupDescription: 'Learn how to protect your EC2 Instance.'
          VpcId: !Ref VPC
          Tags:
               - Key: Name
               Value: 'AWS in Action: chapter 6 (firewall)'
          # allowing inbound ICMP traffic
          SecurityGroupIngress:
               - IpProtocol: icmp
               FromPort: '-1'
               ToPort: '-1'
               CidrIp: '0.0.0.0/0'
```

/chapter06/firewall2.yaml in

Allowing SSH traffic

Once you can ping your EC2 instance, you want to log in to your virtual machine via SSH. To do so, you must create a rule to allow inbound TCP requests on port 22.

```
SecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
        GroupDescription: 'Learn how to protect your EC2 Instance.'
        VpcId: !Ref VPC
            Tags:
            - Key: Name
            Value: 'AWS in Action: chapter 6 (firewall)'
        # allowing inbound ICMP traffic
        SecurityGroupIngress:
            - IpProtocol: icmp
            FromPort: '-1'
            ToPort: '-1'

            CidrIp: '0.0.0.0/0'
            # allowing inbound SSH traffic
            - IpProtocol: tcp
            FromPort: '22'
            ToPort: '22'
            CidrIp: '0.0.0.0/0'
```

Other things to consider:

Allowing SSH traffic from a source IP address

Allowing SSH traffic from a source security group

Bastion (jump host) pattern

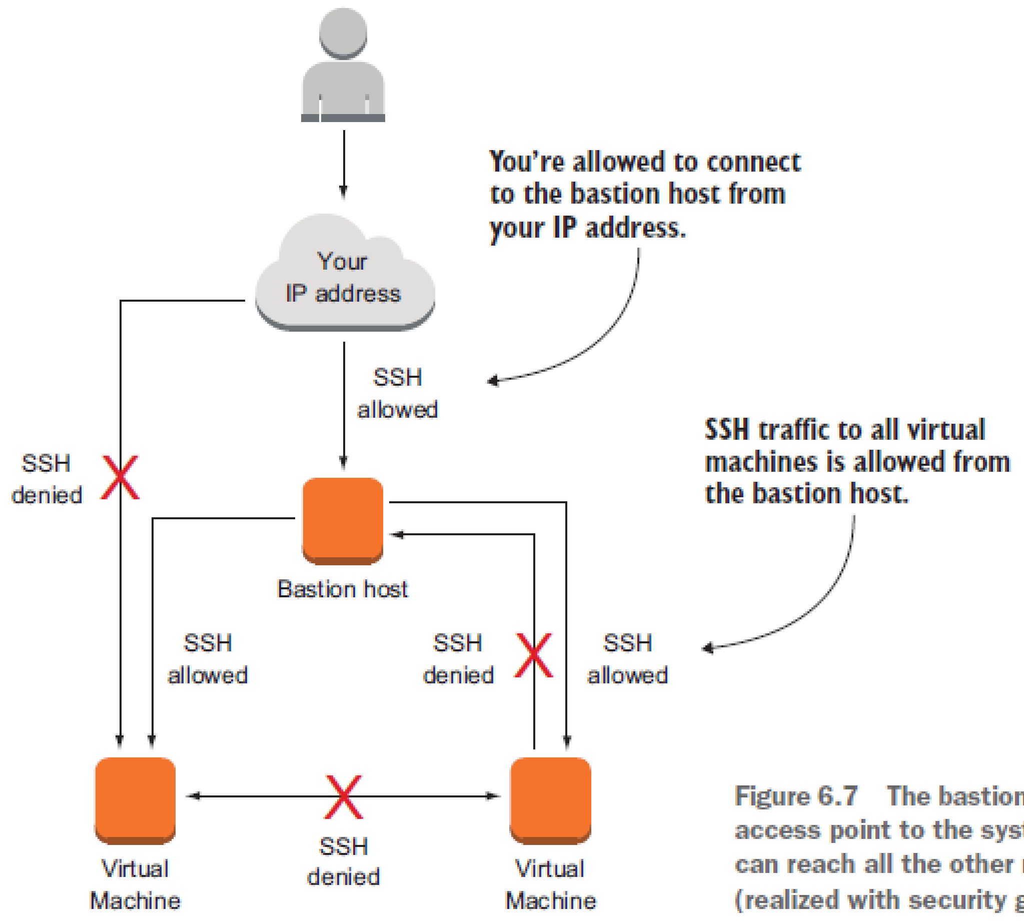

You're allowed to connect to the bastion host from your IP address.
Your IP address
SSH allowed
SSH traffic to all virtual machines is allowed from the bastion host.
SSH denied
Bastion host
SSH allowed
SSH denied
SSH allowed
Virtual Machine
SSH denied
Virtual Machine

Figure 6.7 The bastion host is the only SSH access point to the system from which you can reach all the other machines via SSH (realized with security groups).

Need agent forwarding enabled (potential security risk) or shh proxy for getting ssh access to all systems

```
ssh -J ec2-user@BastionHostPublicName ec2-user@Instance1PublicName
```

/chapter06/firewall5.yaml in

Amazon Virtual Private Cloud (VPC))

your own private network on AWS. *Private* means you can use the custom address ranges
to design a network that isn't necessarily connected to the public internet. You can create subnets,
route tables, ACLs, and gateways to the internet or a VPN endpoint.

Subnets:

Rule of thumb: at least two subnets:

One **Public** and One **Private**.

A public subnet has a route to the internet; a private subnet doesn't.
Your load balancer or web servers should be in the public subnet, and your database should
reside in the private subnet.

ACL – Access Control List

Network ACLs restrict traffic that goes from one subnet to another, acting as a firewall.
That's an additional layer of security on top of security groups, which control
traffic to and from a virtual machine. ACLs behave differently than Security groups. Security groups are
stateful and allow outgoing traffic on allowed incoming ports and ephemeral ports. ACLs do not and need to
be configured. Also, ACL rules have a priority (smaller number indicated higher priority).

IGW – Internet Gateway

The IGW will translate the public IP addresses of your virtual machines to their private
IP addresses using network address translation (NAT). All public IP addresses used in
the VPC are controlled by this IGW.

Route Table

Create specific routes to allow access for private subnets to public resources. For example, allow a server in a private subnet to access the Internet by creating a route to the NAT gateway.

/chapter06/vpc.yaml.

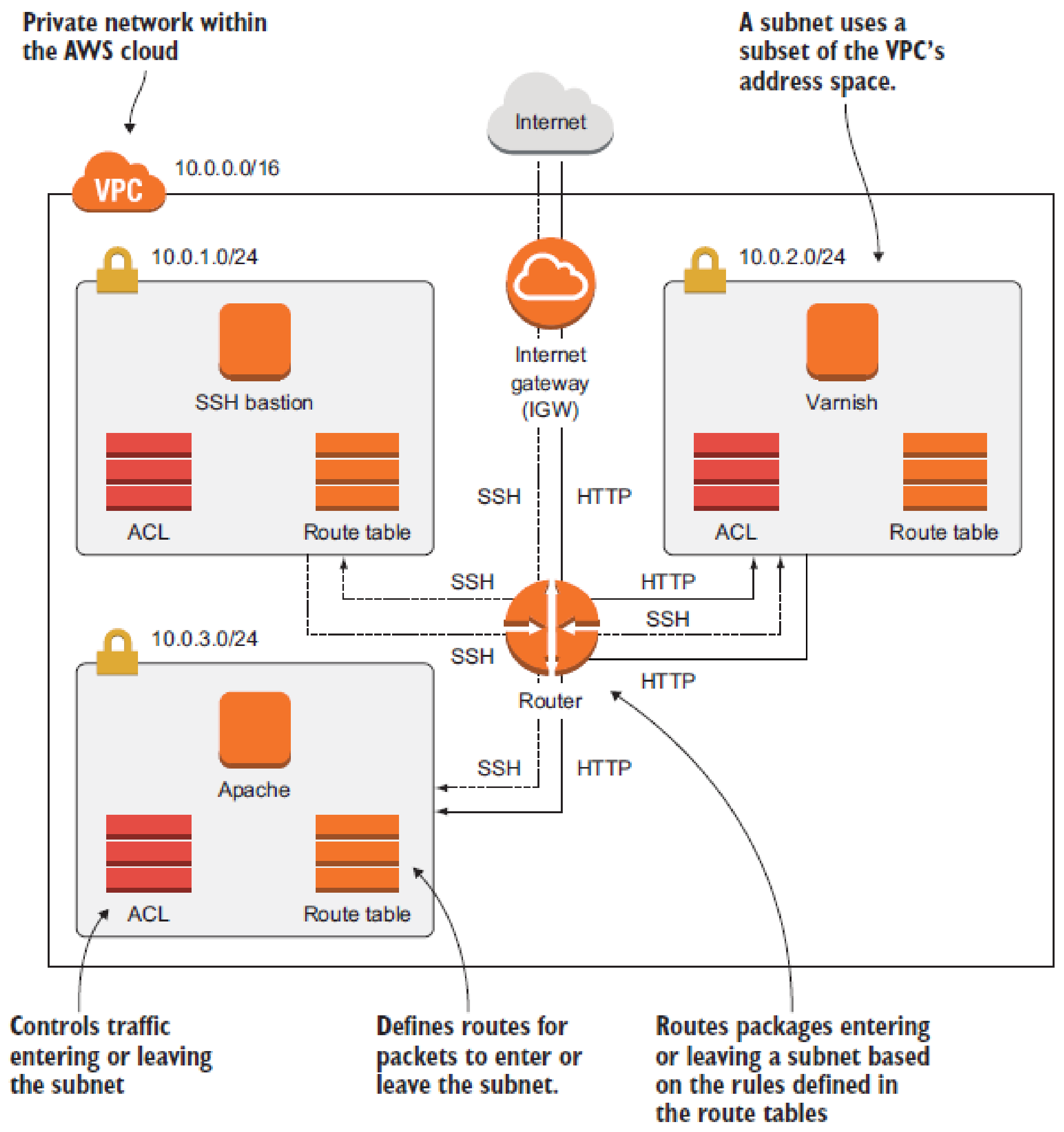

Private network within the AWS cloud
A subnet uses a subset of the VPC's address space.
Internet
VPC
10.0.0.0/16
10.0.1.0/24
SSH bastion
ACL
Route table
Internet gateway (IGW)
10.0.2.0/24
Varnish
ACL
Route table
SSH
HTTP
SSH
HTTP
SSH
SSH
HTTP
10.0.3.0/24
Router
Apache
SSH
HTTP
ACL
Route table
Controls traffic entering or leaving the subnet
Defines routes for packets to enter or leave the subnet.
Routes packages entering or leaving a subnet based on the rules defined in the route tables

Figure 6.8 VPC with three subnets to secure a web application