# Creating docker instance in AWS EC2 instance

After creating a EC2 Linux instance in AWS, follow the below linux commands

1. sudo yum update –y (For linux updates)
2. sudo yum install –y docker (Installing docker setup)
3. sudo groupadd docker (for non root users to avoid permissions, add docker group)
4. sudo usermod –a -G docker $USER (adding current user to docker group, a means add, G means group and its name then user name($user gives current user)) Logout and login (restart) to get changes reflected.
5. Docker –version (to know docker installed or not), sudo su (to login as root user), docker info

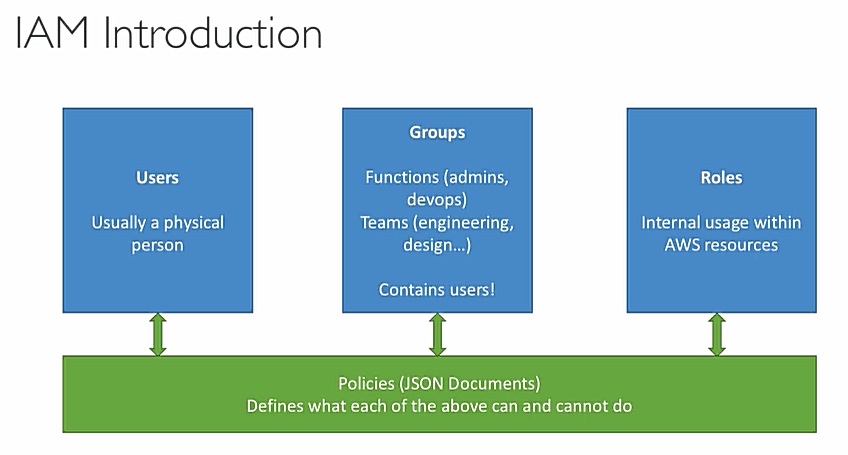
# AWS

## AWS Regions and AZ (Availability Zone)

AWS has regions around the world with names like us-east-1, ap-southeast-2 etc. Each region has Availability zone AZ (us-east-1a,us-east-1b..).Each AZ is a physical data center in the region but separate from other AZ to isolate from disasters. **All AWS consoles are region scoped except the IAM and S3 (these 2 are global services rest all require AZ).**

## IAM (Identity and Access Management)

Whole AWS security users, groups, roles are maintained here. Root account should not be used. All services will use IAM so it needs to be properly maintained. **Policies are written in JSON.** As discussed above IAM has global view, can setup Multi Factor Authentication(MFA) for users. Has predefined policies also like admin etc. or we can create ours.



Big enterprises integrates their own repository of users with IAM, this way they can login into AWS with company credentials (like we login into outlook using company email but not microsoft). This is called IAM Federation. This identity federation uses the SAML standard (security assertion markup language which enables SSO (single sign on) authentication by passing authentication credentials to service providers) (active directory).

**Best practices:** One IAM User per physical person, One IAM role per application, Don’t write IAM credentials in code, never use root account account for anything except initial setup, never share the credentials.

### Security Credentials

**Password** to login to account, **MFA** to generate code during login, **Access Keys** – To make programmatic calls to AWS from AWS CLI, powershell, AWS SDKs, API calls etc. Can have maximum 2 access keys.

**CloudFront Key pairs** – we can configure key value pairs to use them anywhere inside EC2, like create a key called name and give value of your name and you can use that variable everywhere. If we want to change name we can change the variable directly

**X.509 certificate** – These certificates are used to make secure SOAP-protocol requests to AWS services like S3 etc.

**Account Identifiers** – AWS account ID a 12 digit account ID to refer our account programmatically and in other contexts and a Canonical User ID to configure S3 access controls lists.

To create a new IAM user, we give their username, access permissions, roles like administrator access etc (we can define these roles through groups), their password defined. However we directly should not assign roles and permissions to each user individually, so we create groups with their permissions like admin group and what permissions they have inside that group then we will assign that group to all the admin users. Next is IAM password policy in which we can set the password policy (like length, complexity etc.) of the users. So by this almost all 5 steps are green in IAM and we are good to go. Edit our account login URL and change the alias from account ID number to anything unique and easy to access. Generally alias will be org name so the sign in URL looks like eg: <https://infosys.signin.aws.amazon.com/console> then we will login with created IAM user credentials. Keep the root account aside and use the created IAM user from now.

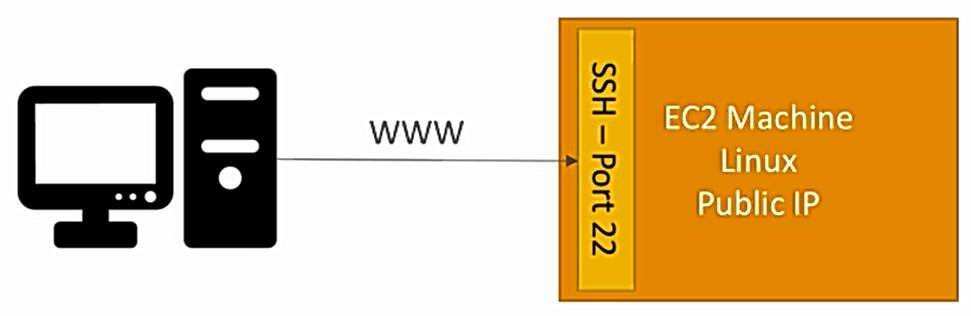
## EC2

Renting Virtual Machine (EC2), it consists following services also - Storing data on virtual drives (EBS), Distributing load across machines (ELB), Scaling the services using auto-scaling groups (ASG)

We can launch EC2 mirco instance with linux or windows OS, start/stop them.

To create a EC2 image, first we select the Amazon machine image (linux,windows,redhat,…which some are free tier ) then we will select the type of instance that is t2 micro, no of virtual CPUs, memory, network performance, whether EBS is enabled or not etc. Then we configure instance by checking up options like network which is default vpc, subnet (AZs) and then under advanced options we have User Data which means commands to run at instance start (discussed deeply below), IAM role follow or not etc. Then proceed to storage details which is EBS volume like amount of storage, type of storage to be used etc. then in next step we can define tags which are key value pairs to be used inside EC2 instance, name tag is important cause it shows in UI. Then we move to security which acts like firewall, we can create a new security group or create a new one there by giving type, protocol, source(allowable IP, 0.0.0.0/0 means all IPs) etc. then we review and launch. Finally it asks for a key-pair (.pem file) which is of 2 types private and public. Public is for aws stores and private is for user to store so that to connect instance securely like password to login,SSH to allow all will be done by these key-pairs match. So pem file can be downloaded only once during creation of instance and need everytime if we want to connect to the instance.

To use the launched ec2 instance, we access it through SSH

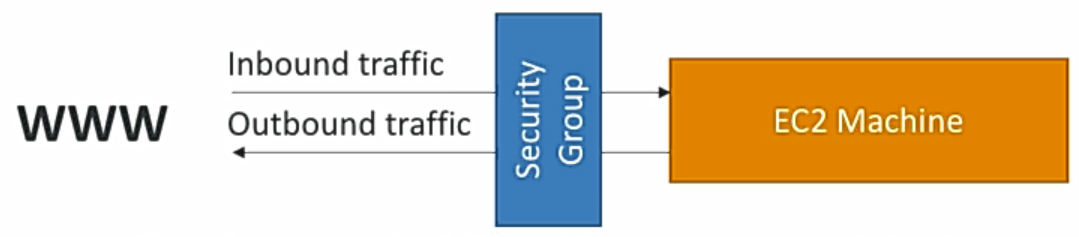


SSH (Secure Shell) is used to connect and control our instance or remote machine using cli. In linux and mac, we will do directly in command line bu doing ssh commands (ssh –i EC2Tutorial.pem [ec2-user@56.189.64.15](mailto:ec2-user@56.189.64.15) , this command will login into our instance) where as in windows we use **Putty (open source SSH client).**

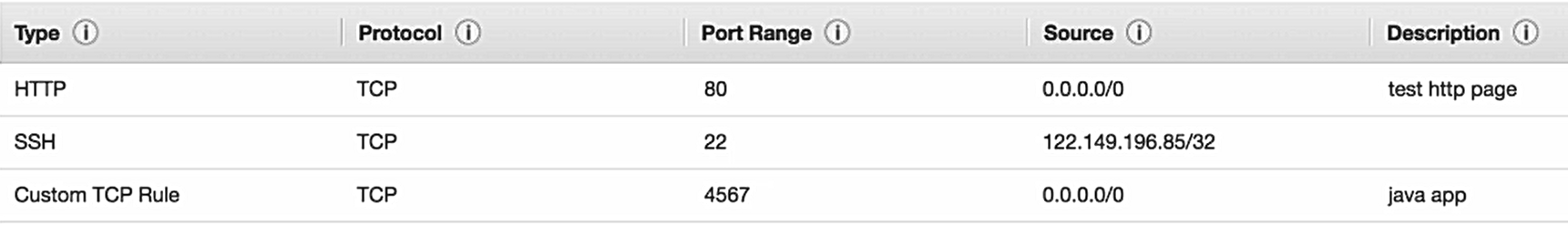
Download putty and install, then open puttygen to convert our .pem file into .ppk file so that putty can read it, just load pem file and save it as ppk. After that open putty and give host IP as the ec2-user @ IP4 address of our instance

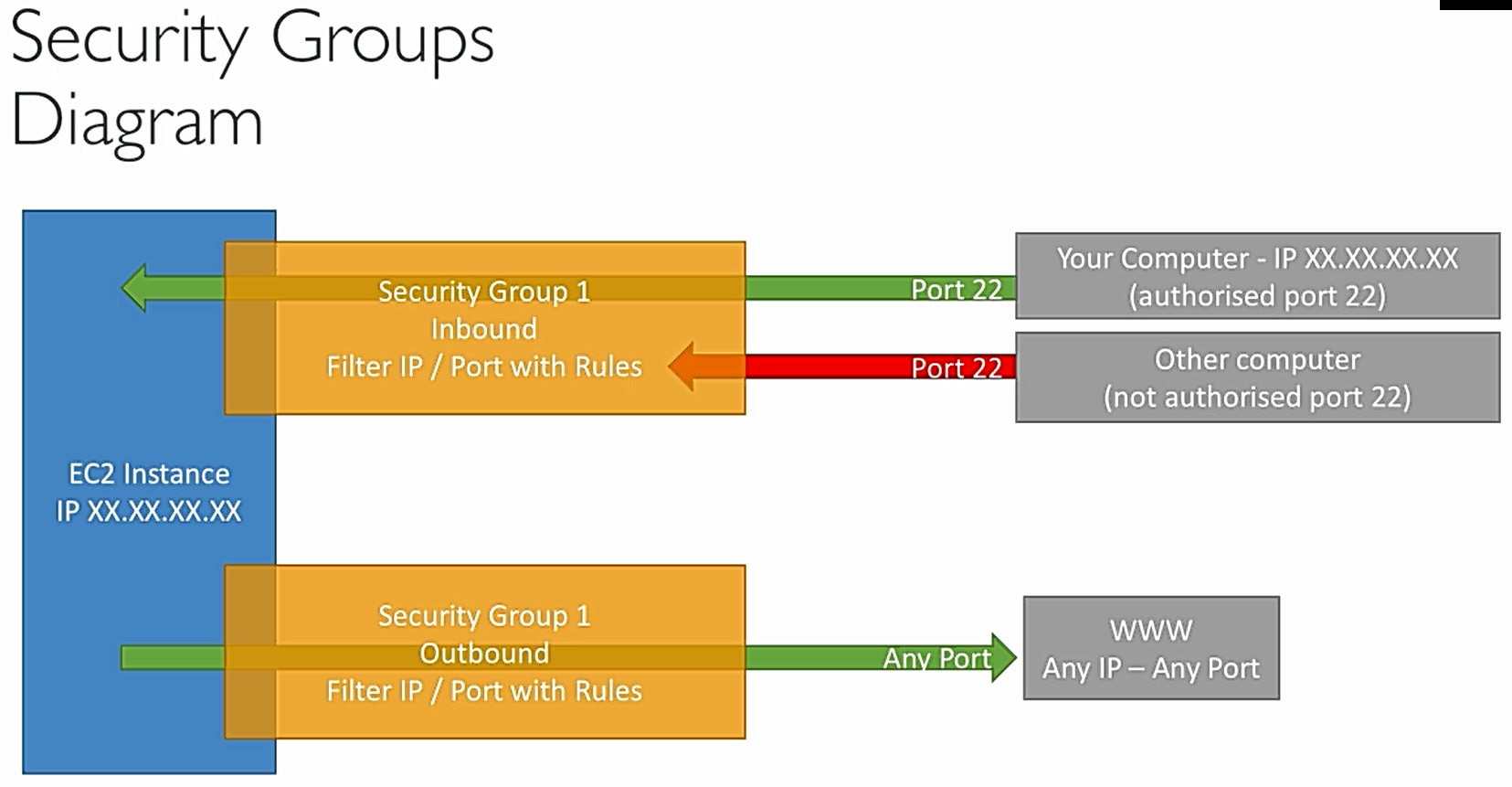
Eg: ec2-user@56.189.64.15 , to make it use of ppk file generated, go to SSH under connection on left side navigation and oAuth in it. Give path of ppk file. Save the entire session and save it so that we can use it multiple times, no need to setup everytime. So after successful connection we will connect to instance in cli and run cmds eg:whoami etc.

## Security Groups (SGs)

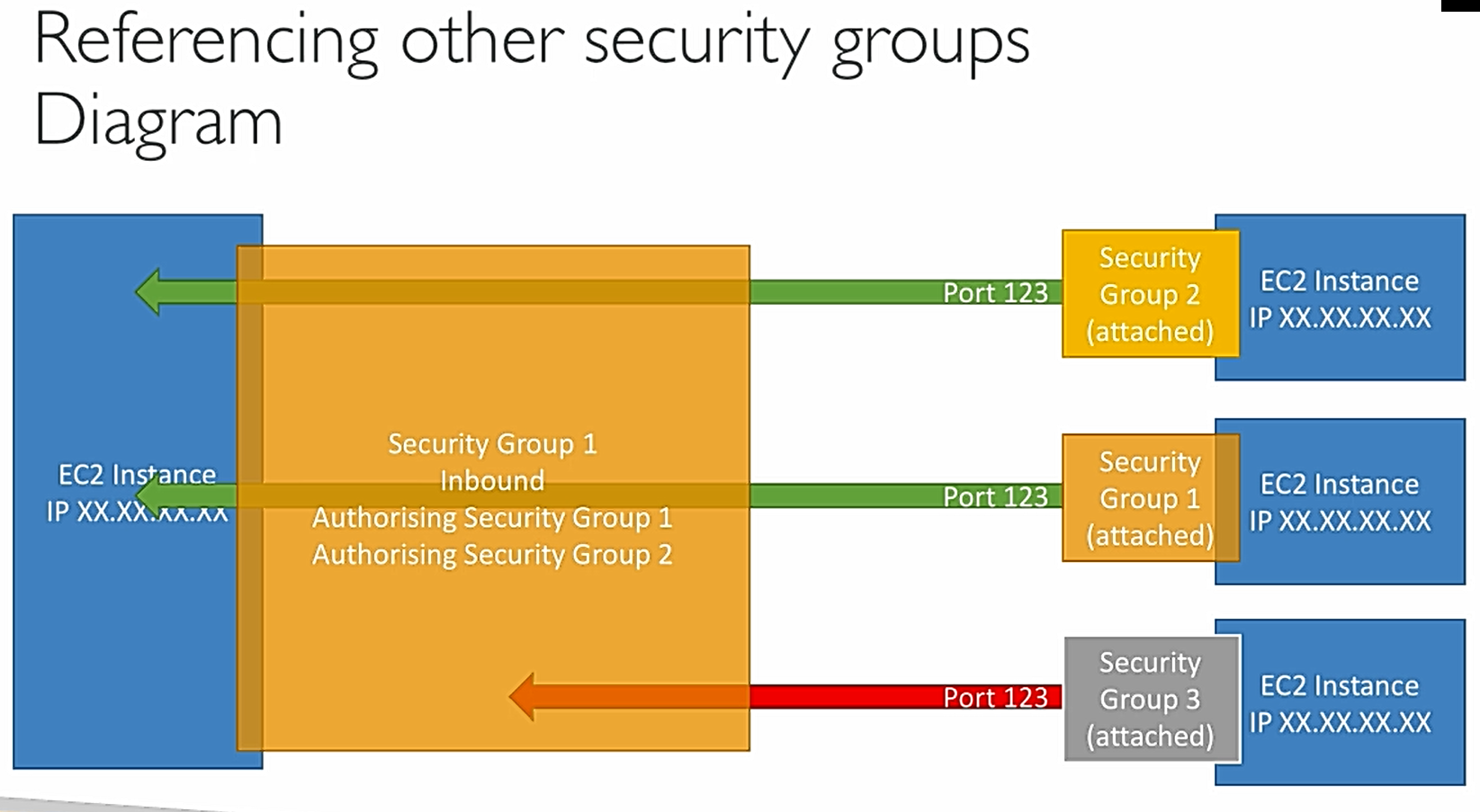


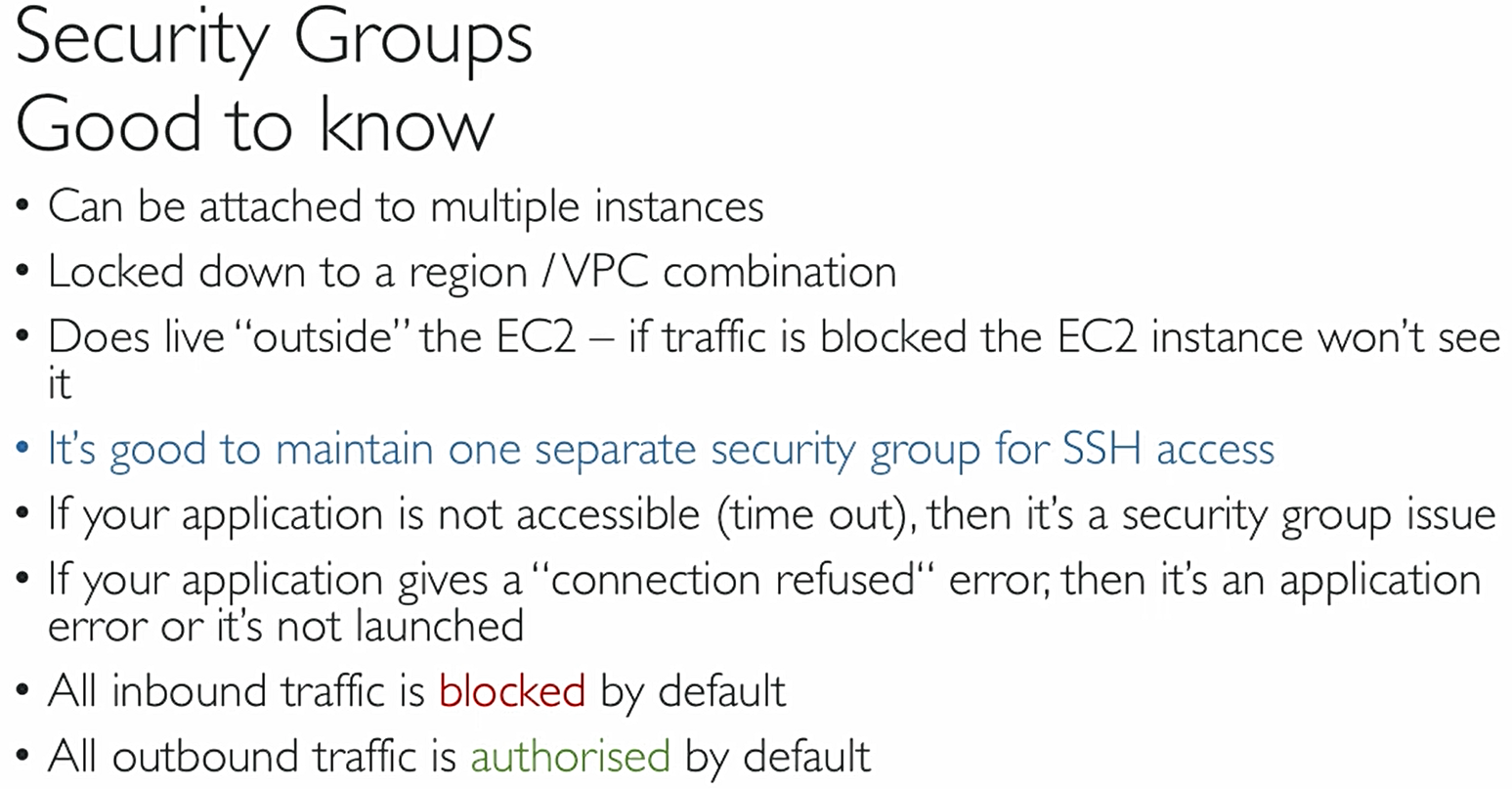
Acts as firewall of the EC2 instance. They control inbound and outbound traffic, basic feature of cloud, fundamental networking skill to troubleshoot networking issues. So defined inbound rules are not met then we get timeout error if we try to connect to instance. We can create a security group in networking and security section in AWS by giving Type like SSH,HTTP or HTTPs etc., protocol, allowable port range, source like only our IP or matching IP or 0.0.0.0/0 to allow all for IPv6 0.0.0.0/0::0 and then finally description of security group. Like this we can define outbound (instance to outside) rules too.



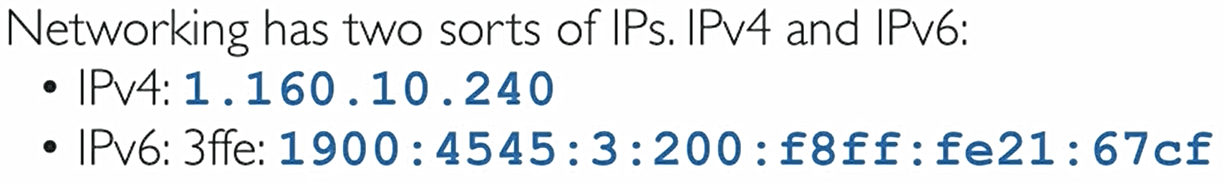


We can also allow traffic if security group matches directly without matching the rules stated. Like in a security group just like how we define allowable IPs etc. we can define allowable other security groups. We will see this in load balancing. Refer below.



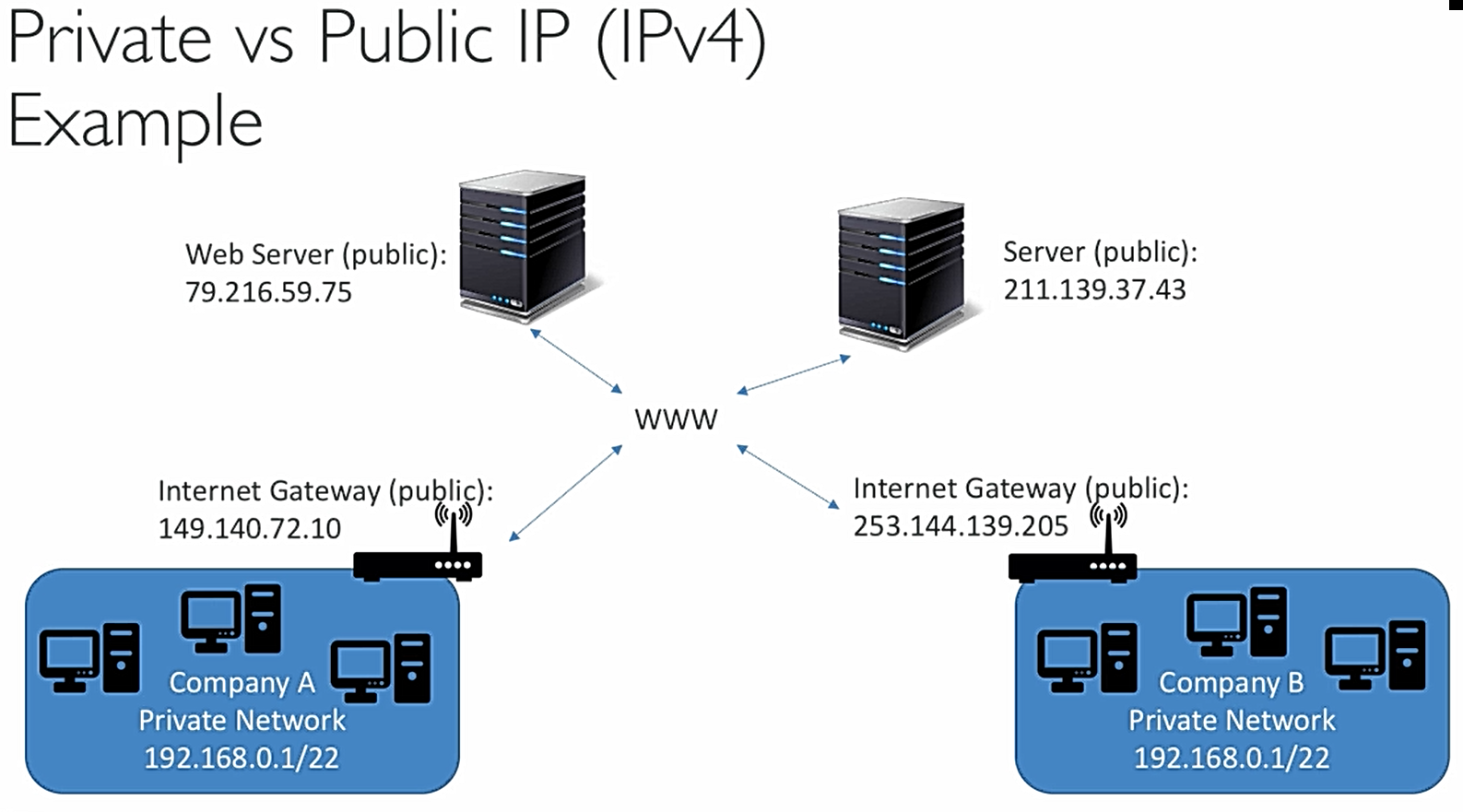


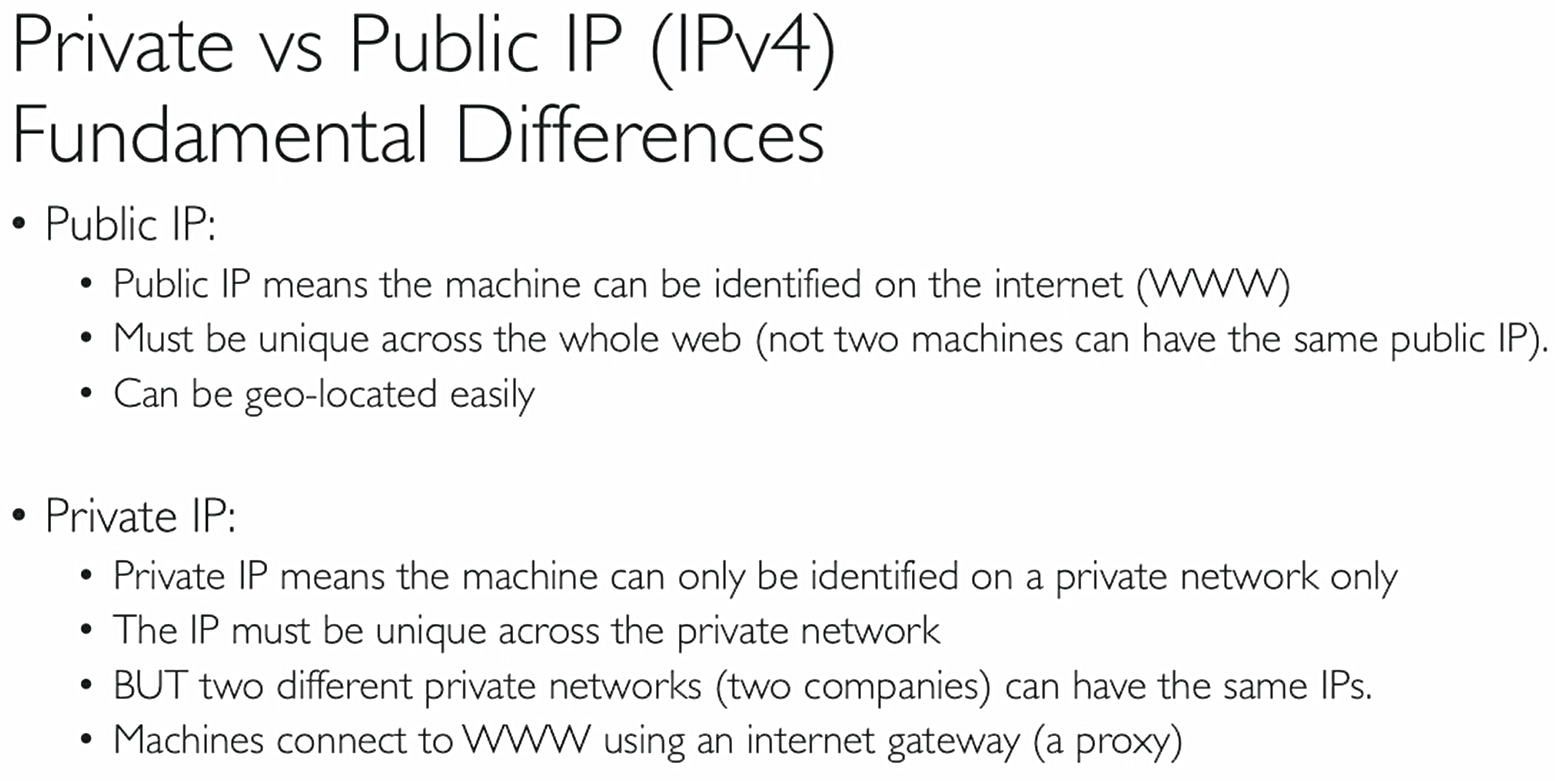
## Networking Public vs Private Elastic IP



IPv4 is most common format in internet [0-255]. [0-255]. [0-255]. [0-255], IPv6 is newer and used in Iot etc.

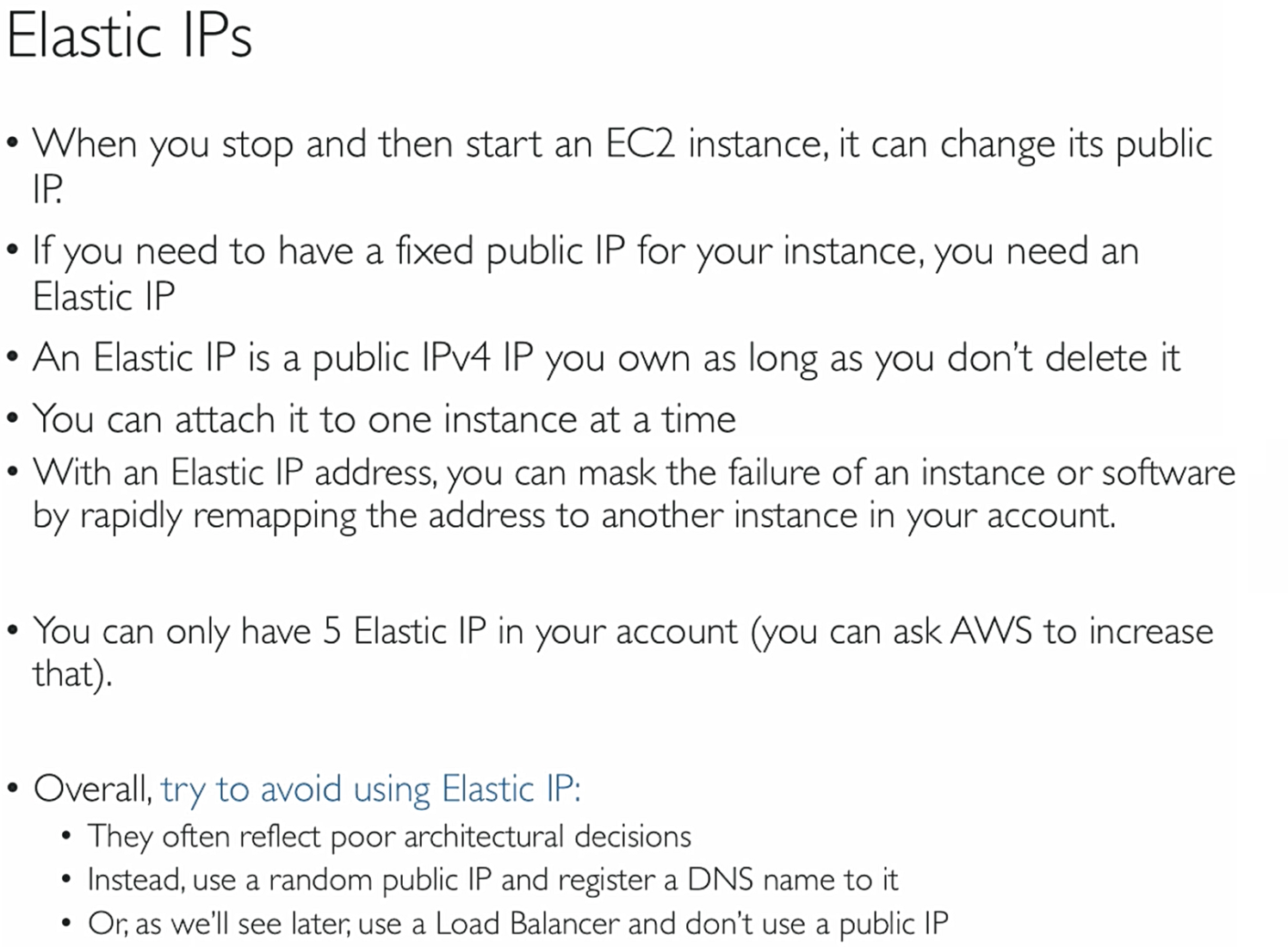
When it is public network then we can directly access it through internet whereas if it is private IP then we can access only within that private network, we need internet gateway which is public to access internet.

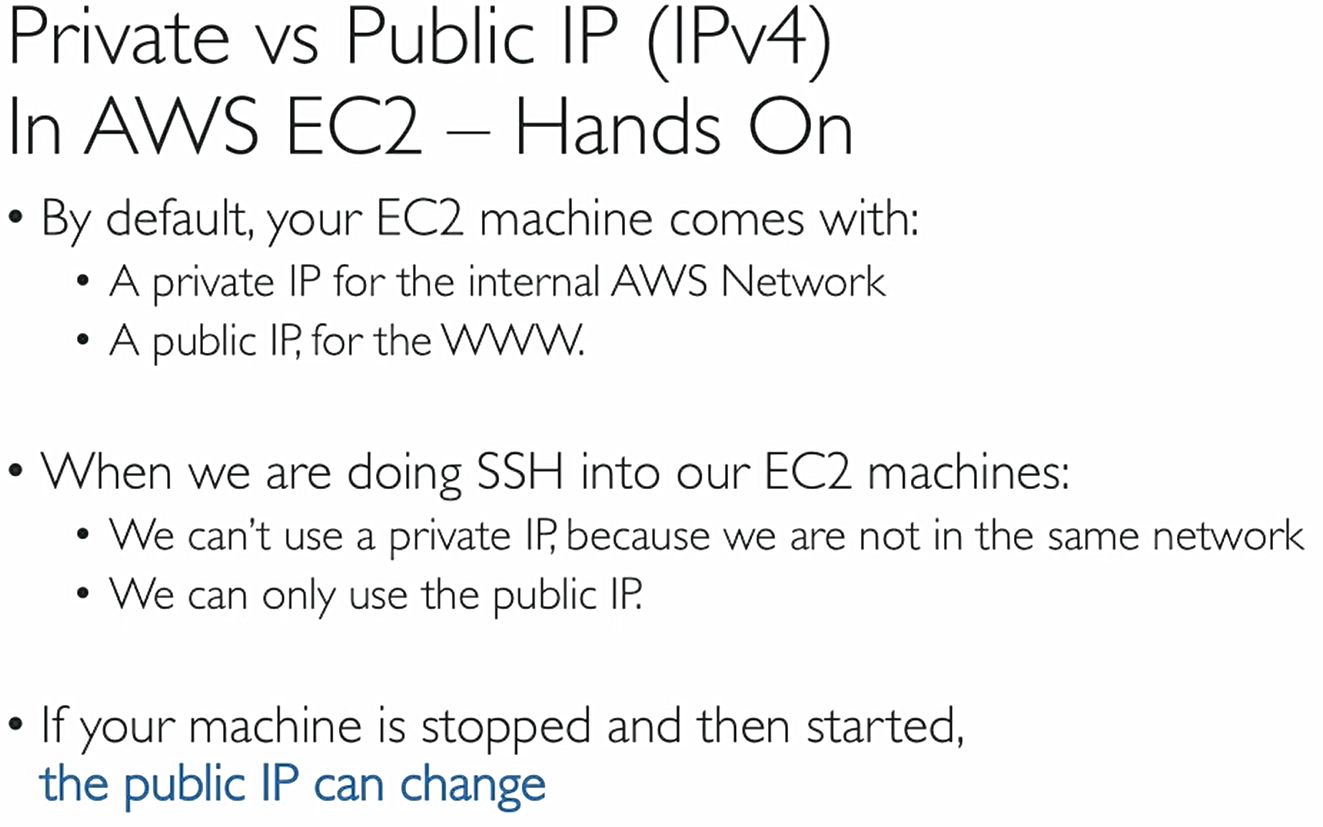




Since every time public IP changes when we restart the ec2 instance, we need to update the ssh command or putty host ip accordingly with new public ip generated, however throughout the instance life cycle the private IP remains same cause it is there in AWS network. We cannot use private IP to connect (timeout error) cause we are not in the same network but once we connect through public IP there we can see in console of putty ec2-user@private IP address cause we are connected to our instance and we are inside AWS network we can see private IP

Regarding the elastic IP (found under Network and security group), create a new elastic IP and we will associate our ec2 instance to it and now if we see the info our instance the public IPv4 address the auto generated one is replaced with the link of elastic IP address. This elastic IP is bound to life cycle of instance just like private IP and doesn’t change even if we restart the instance. We can disassociate by right click options of instance.



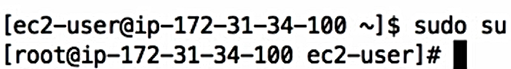


## Operations on Instance

In this we practise by installing apache webserver to display webpage and creating a html file to display hostname.

Connect to the instance (make sure it is Amazon linux 2 AMI(Amazon machine image)) through Putty or ssh command with latest public IP and then run following commands (“clear” command will clear screen)

1. sudo su :-Gives root access to the user, so that we cann run all commands without asking for permissions



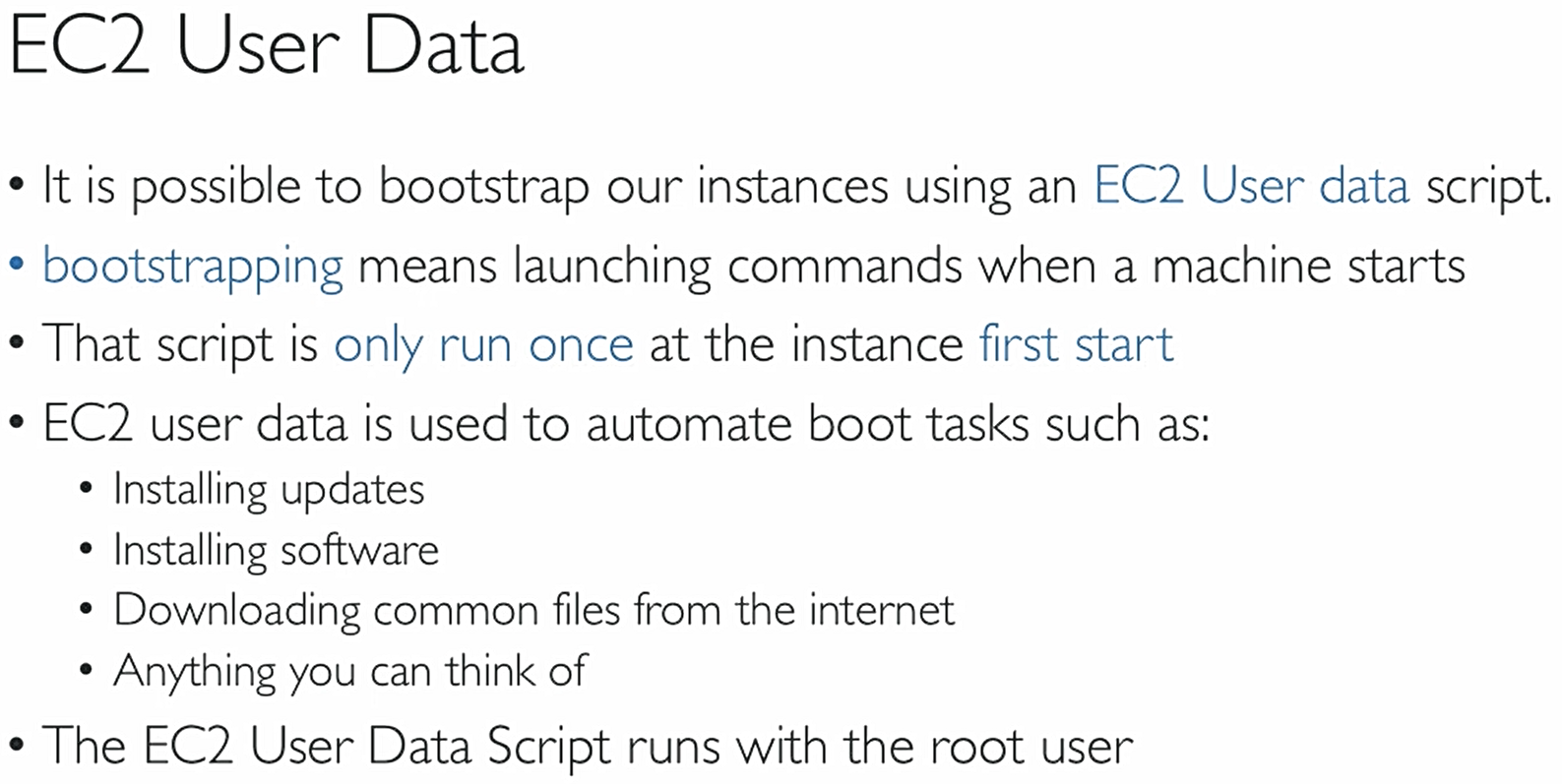
1. yum update –y :- this command will update all packages in machine to the latest versions, -y mean yes to all
2. yum install –y httpd.x86\_64 :- install apache server httpd of 86,64 bit and –y yes to all permissions
3. systemctl start httpd.service :- starting the httpd service, systemctl command found in amazon linux 2 only
4. systemctl enable httpd.service :- above command starts and this will enable the service across the reboots
5. curl localhost:80 :-curl means current url and then url this command loads the localhost:80 webpage html which is apache homepage in cli cause apache runs on localhost default 80 port always.
   1. We can see this apache homepage in our browser by url our public IPv4 followed by apache port or only public ip, so kind of looks like 34.45.33.54:80 (assume it’s a public IP). But this will result in time out error.
   2. That’s because in our security group we mentioned by default to use SSH access that too only on 22 port. So enable connection edit the Security group and create a new rule with HTTP connection and port as 80, IP as 0.0.. as of now and reload the previous mentioned url, this time without timeout error the apache homepage will appear which is running in our ec2 instance. If we look carefully, we find



So to appear our stuff instead of apache default homepage on this port we need to edit /var/www/html/. File, the .file is generally index.html

1. echo “hello pavan” > /var/www/html/index.html :-echo command will wirte the content “hello pavan” inside(>) the file of given path. So if we reload the public ip4:80 we see Hello pavan html there. To get the host name command is “hostname –f” this will give ip.AZ… so to display this inside html
2. echo “hello pavan from $( hostname –f)” > /var/www/html/index.html :-shows hello pavan from host details

## EC2 User Data



So to automate the apache server and html file edit which we did in above step we give the same commands in user data file. Create a new instance(we can also stop existing one and give user data by right click options) and in user data options repeat following commands and in security group make sure to allow HTTP of port 80 or simply use the existing sg we created above earlier to allow http port 80

we start with #!/bin/bash that means we are stating that use only bash to run the following commands, no other shell. Since all commands run on root user mode, no need to write sudo su command. Also before copying below code look for – it will copy differently so edit them

#!/bin/bash

yum update -y

yum install -y httpd.x86\_64

systemctl start httpd.service

systemctl enable httpd.service

echo “hello pavan from $( hostname -f)” > /var/www/html/index.html