BHARATI VIDYAPEETH’S
INSTITUTE OF COMPUTER APPLICATIONS &
MANAGEMENT (Affiliated to Guru Gobind Singh Indraprastha
University, Approved by
AICTE, New Delhi)

Cloud Computing (MCA-203)
Practical File

Submitted To:
Dr. Arpita Nagpal
Assistant Professor

Submitted By:
Saurabh Kumar
04211604422
MCA III (Sec A)

INDEX

P1: a) Assume you have started your own entrepreneur and your work is increasing at a high speed, you employ more workers. Now you will take the help of cloud providers. Give the details of different providers. Emphasis upon: P-cloud, Mega Cloud, google cloud, Aws, Azure.
b) Describe Google app engine also.

Ans:
1. Amazon Web Services (AWS):
- Offers a wide range of cloud computing services, including computing power, storage, and databases.
- Scalable and flexible infrastructure for businesses of all sizes.
- Popular services include EC2 (Elastic Compute Cloud), S3 (Simple Storage Service), and RDS (Relational Database Service).

2. Microsoft Azure:
- Provides a comprehensive suite of cloud services, including virtual machines, databases, AI, and analytics.
- Offers integration with Microsoft products and tools.
- Services like Azure Virtual Machines, Azure App Service, and Azure SQL Database.

3. Google Cloud Platform (GCP):
- Offers cloud computing, storage, machine learning, and data analytics services.
- Known for its data analytics and AI capabilities.
- Services like Google Compute Engine, Google Cloud Storage, and BigQuery.

4. IBM Cloud:
- Provides a variety of cloud services, including computing, storage, and AI.
- Offers hybrid and multi-cloud solutions.
- Services like IBM Virtual Servers, IBM Cloud Object Storage, and Watson AI services.

5. Oracle Cloud:
- Focuses on database, enterprise applications, and cloud infrastructure.
- Strong offerings for businesses using Oracle software.
- Services like Oracle Compute, Oracle Database Cloud, and Oracle Cloud Infrastructure.

6. Alibaba Cloud:
- A leading cloud provider in China and Asia, with global expansion.
- Offers a wide range of cloud services, including computing, data storage, and networking.
- Services like Elastic Compute Service (ECS), Object Storage Service (OSS), and MaxCompute.

7. DigitalOcean:
- Known for simplicity and developer-friendly features.
- Provides virtual private servers (droplets) and managed Kubernetes clusters.

- Services like Droplets, Spaces (object storage), and Managed Databases.

8. Salesforce:
- Specializes in customer relationship management (CRM) and business applications.
- Offers cloud-based solutions for sales, service, marketing, and more.
- Services like Salesforce Sales Cloud, Service Cloud, and Marketing Cloud.

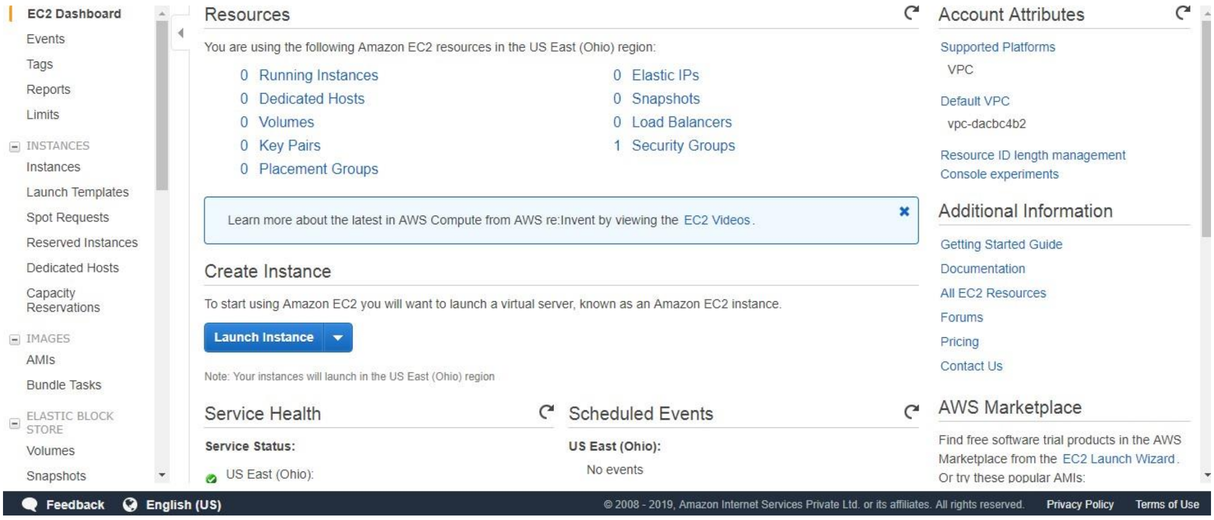
9. VMware Cloud:
- Focuses on virtualization and cloud infrastructure.
- Offers solutions for running virtualized workloads in the cloud.
- Services like VMware Cloud on AWS and VMware Cloud Foundation.

10. Red Hat OpenShift:
- An enterprise Kubernetes platform for containerized applications.
- Focuses on hybrid cloud and multi-cloud environments.
- Offers features for deployment, scaling, and management of containerized applications.

Google App Engine (GAE) is a fully managed Platform as a Service (PaaS) for developing and hosting web applications at scale. It uses the same infrastructure as Google’s large-scale internet services.
Applications hosted on GAE are sandboxed and run across multiple servers for redundancy, allowing for scaling of resources according to the traffic requirements of the moment. This means that your applications can dynamically scale as demand changes over time.
GAE supports several popular languages, libraries, and frameworks. You can develop your applications in languages such as Java, Python, PHP, Go, Node.JS, .NET, and Ruby.
It also provides several built-in APIs that can be used to add features and functionality to your application. For example, the Google App Engine Cloud storage API can be used to store and retrieve data.
In addition to these features, GAE offers two environments: the Standard Environment and the Flexible Environment. The Standard Environment is based on containers running on Google’s infrastructure and supports languages like Python, Java, Node.js, Ruby, PHP, and Go. The Flexible Environment allows more customization options such as running custom runtimes using Dockerfiles.

P2: **a) Find a procedure to set up an account with AWS, services it offers. Support your procedure with screenshots.**
b) *Demonstrate* the steps with screenshots to add a new instance and create a virtual machine in Amazon Web Service using
EC2.

Step 1:



Resources

You are using the following Amazon EC2 resources in the US East (Ohio) region:

0 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
0 Volumes	0 Load Balancers
0 Key Pairs	1 Security Groups
0 Placement Groups	

Learn more about the latest in AWS Compute from AWS re:Invent by viewing the EC2 Videos.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (Ohio) region

Service Health

Service Status:

US East (Ohio):

Scheduled Events

US East (Ohio):

No events

Account Attributes

Supported Platforms
VPC

Default VPC
vpc-dacbc4b2

Resource ID length management
Console experiments

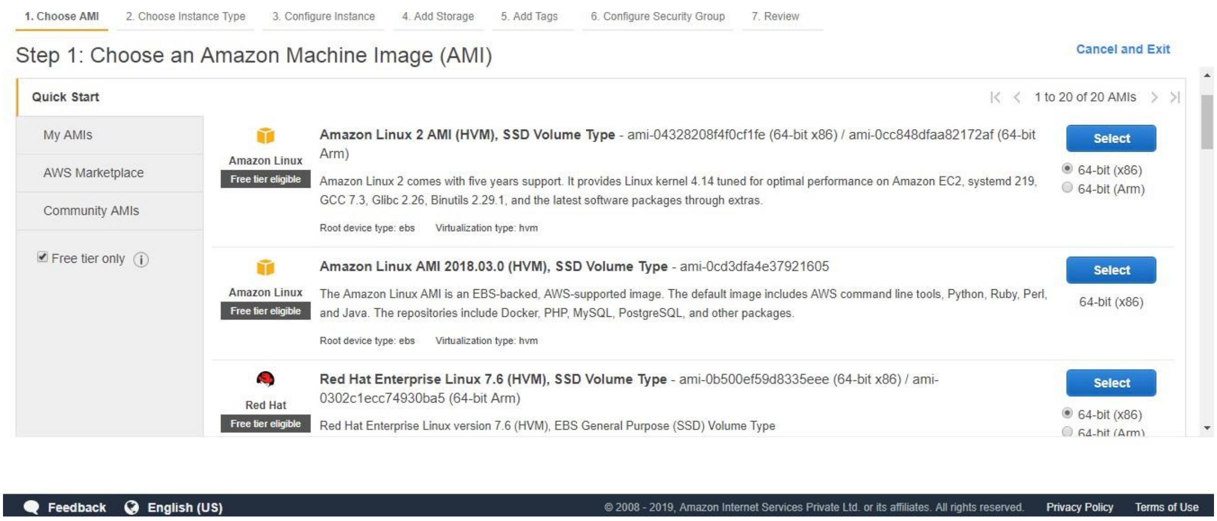
Additional Information

AWS Marketplace

Find free software trial products in the AWS Marketplace from the EC2 Launch Wizard.
Or try these popular AMIs:

© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Step 2:



Step 1: Choose an Amazon Machine Image (AMI)

☑ Free tier only

1 to 20 of 20 AMIs

Amazon Linux
Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-04328208f4f0cf1fe (64-bit x86) / ami-0cc848dfaa82172af (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

Root device type: ebs   Virtualization type: hvm

Select
● 64-bit (x86)
○ 64-bit (Arm)

Amazon Linux
Free tier eligible

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0cd3dfa4e37921605

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root device type: ebs   Virtualization type: hvm

Select
64-bit (x86)

Red Hat
Free tier eligible

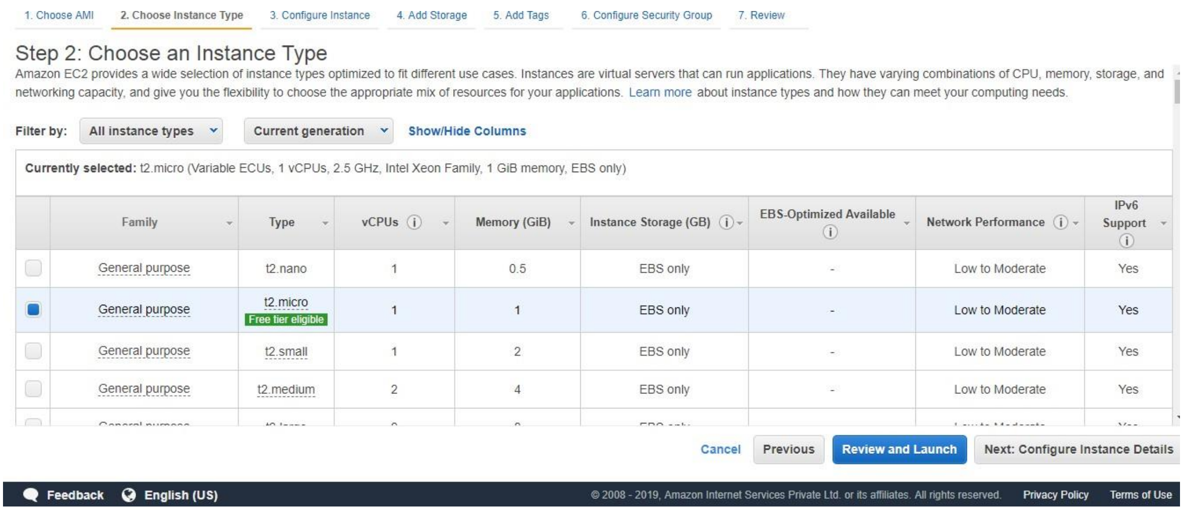
Red Hat Enterprise Linux 7.6 (HVM), SSD Volume Type - ami-0b500ef59d8335eee (64-bit x86) / ami-0302c1ecc74930ba5 (64-bit Arm)

Red Hat Enterprise Linux version 7.6 (HVM), EBS General Purpose (SSD) Volume Type

Select
● 64-bit (x86)
○ 64-bit (Arm)

© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Step 3:



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

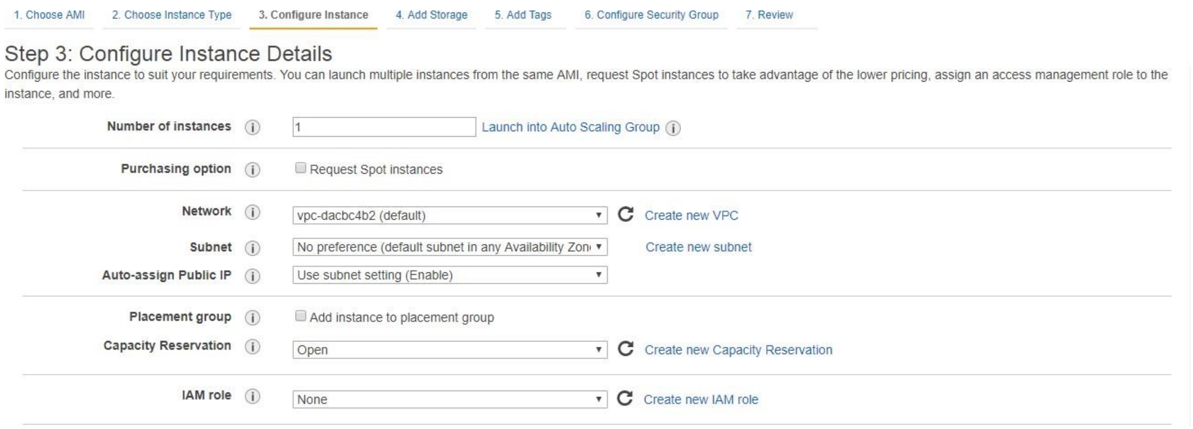
Filter by: All instance types   Current generation   Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
□	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes

© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Step 4:



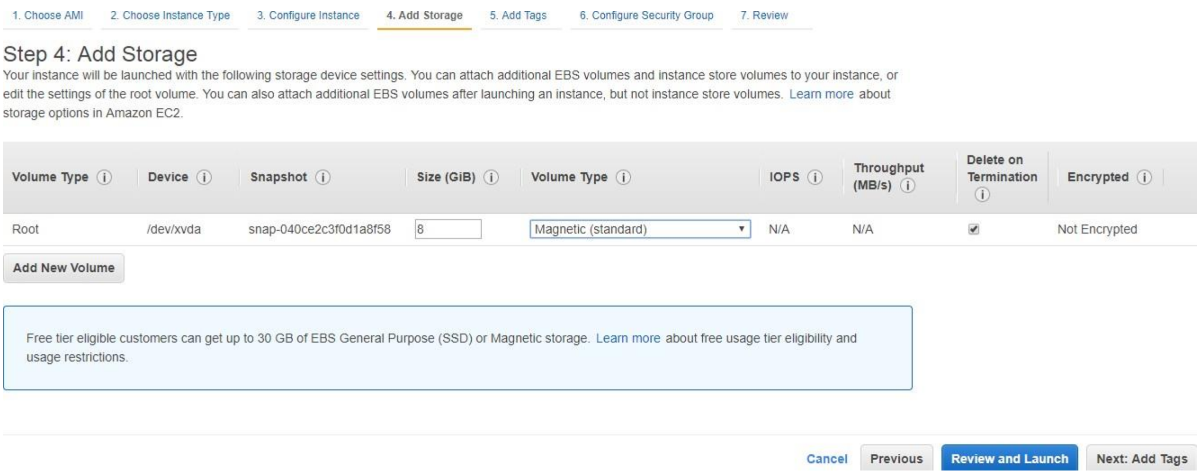
1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances ⓘ	1	Launch into Auto Scaling Group ⓘ
Purchasing option ⓘ	☐ Request Spot instances	
Network ⓘ	vpc-dacbc4b2 (default) ▾	Create new VPC
Subnet ⓘ	No preference (default subnet in any Availability Zon ▾	Create new subnet
Auto-assign Public IP ⓘ	Use subnet setting (Enable) ▾	
Placement group ⓘ	☐ Add instance to placement group	
Capacity Reservation ⓘ	Open ▾	Create new Capacity Reservation
IAM role ⓘ	None ▾	Create new IAM role

Step 5:



1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 4: Add Storage

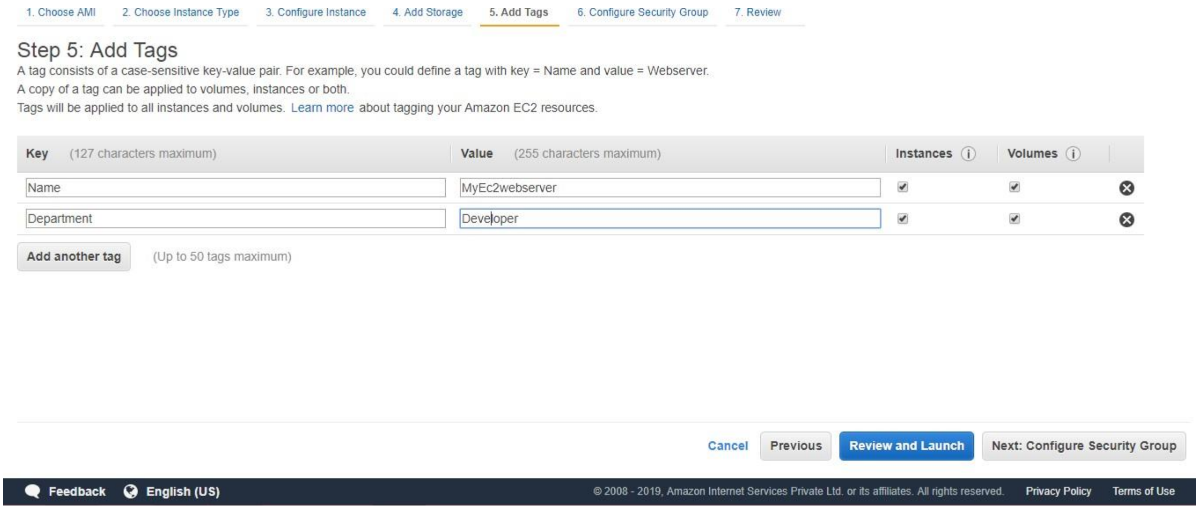
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/xvda	snap-040ce2c3f0d1a8f58	8	Magnetic (standard) ▾	N/A	N/A	☑	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

Step 6:



1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 5: Add Tags

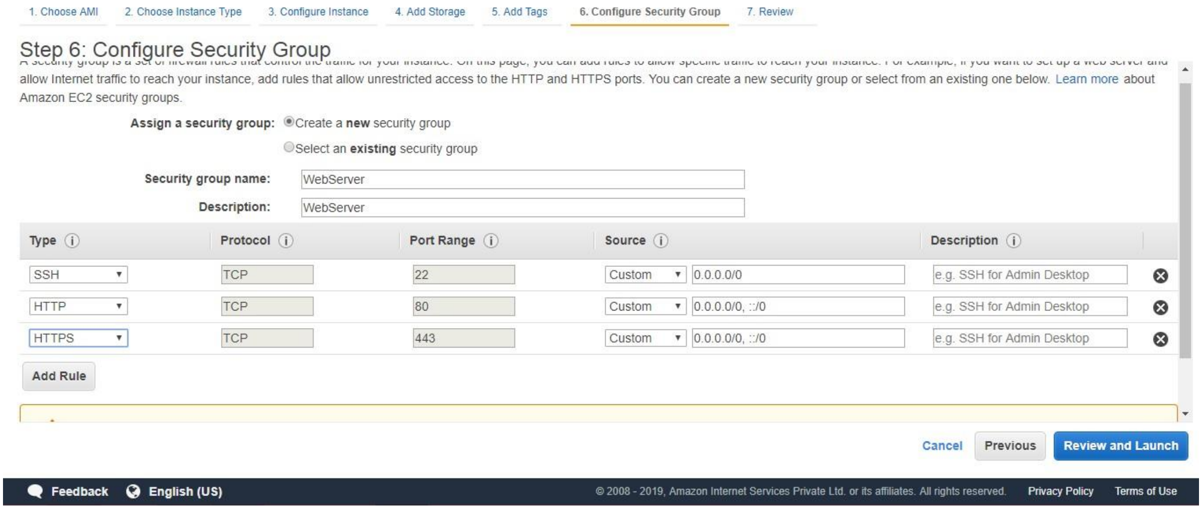
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)	Instances ⓘ	Volumes ⓘ	
Name	MyEc2webserver	☑	☑	⊗
Department	Developer	☑	☑	⊗

Add another tag (Up to 50 tags maximum)

Cancel
Previous
Review and Launch
Next: Configure Security Group
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Step 7:



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ● Create a **new** security group

○ Select an **existing** security group

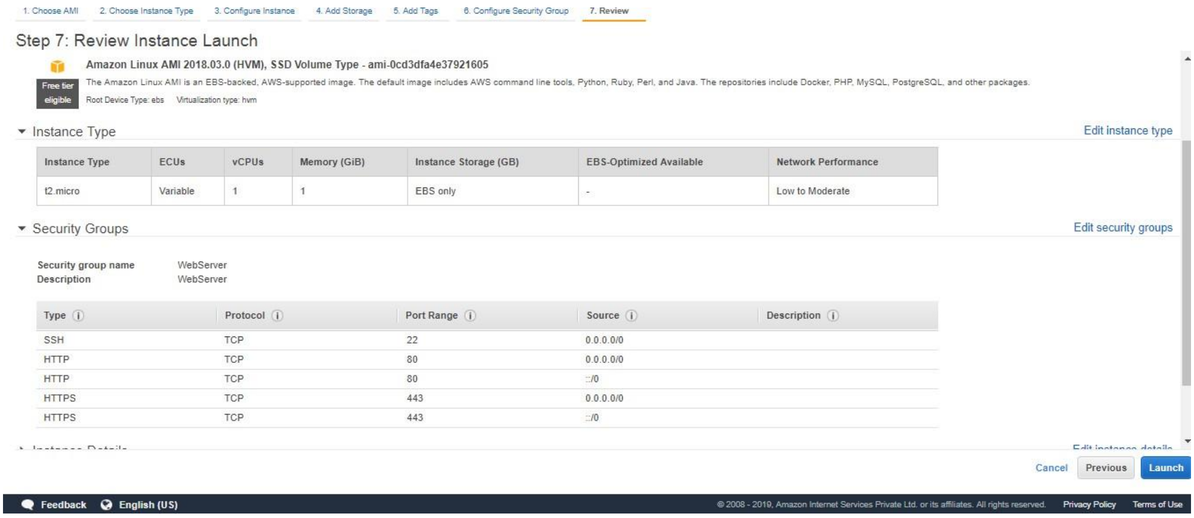
Security group name: WebServer

Description: WebServer

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule
Cancel
Previous
Review and Launch
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Step 8:



Step 7: Review Instance Launch

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0cd3dfa4e37921605

Free tier eligible

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root Device Type: ebs Virtualization type: hvm

Instance Type

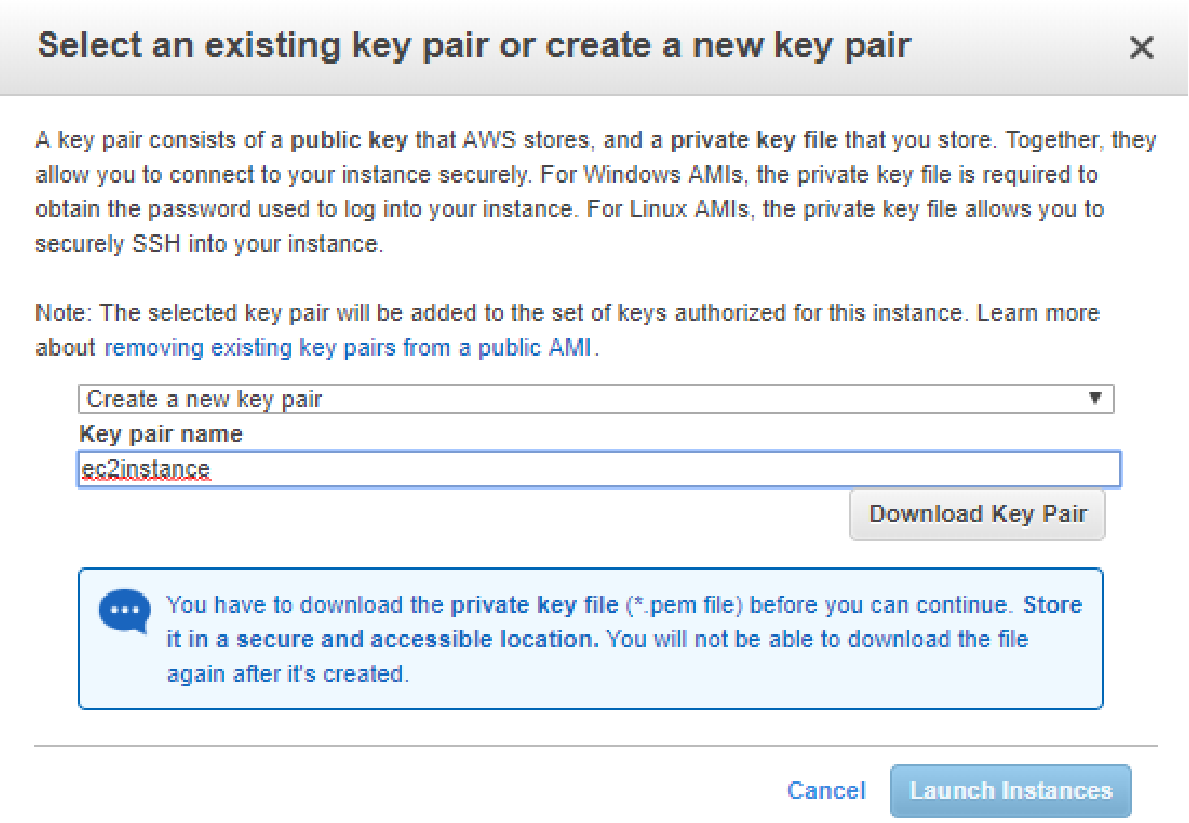
Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

Security group name WebServer
Description WebServer

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	
HTTPS	TCP	443	0.0.0.0/0	
HTTPS	TCP	443	::/0	

Step 9:



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

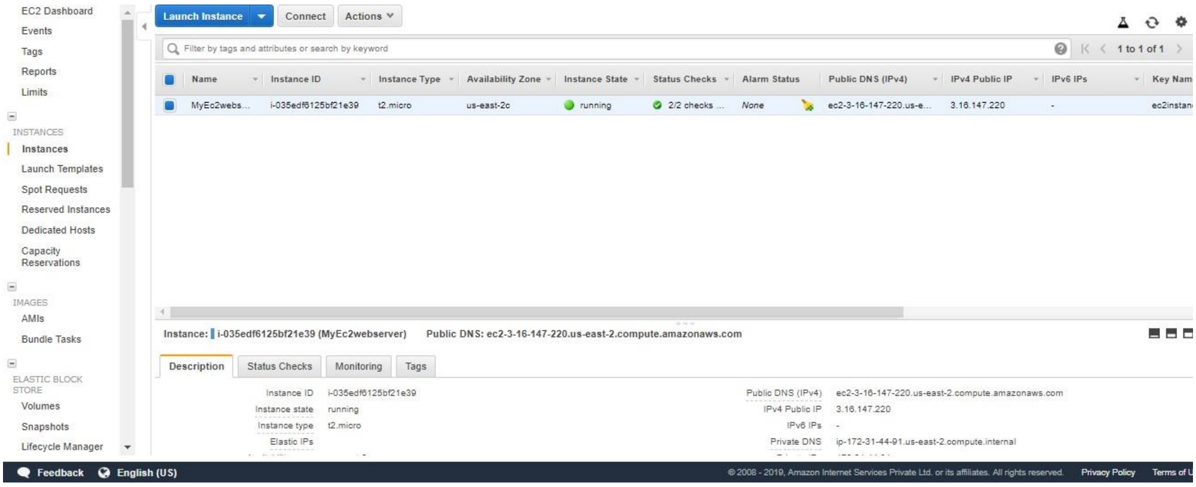
Create a new key pair

Key pair name

ec2instance

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Step 10:



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Nam
MyEc2webs...	i-035edf6125bf21e39	t2.micro	us-east-2c	running	2/2 checks ...	None	ec2-3-16-147-220.us-e...	3.16.147.220	-	ec2instan

Instance: i-035edf6125bf21e39 (MyEc2webserver) Public DNS: ec2-3-16-147-220.us-east-2.compute.amazonaws.com

Instance ID	i-035edf6125bf21e39	Public DNS (IPv4)	ec2-3-16-147-220.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	3.16.147.220
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-44-91.us-east-2.compute.internal

Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of U

P3: Suppose you are an AWS cloud consultant. Mr. Hemant came to you with following constraints:

1. He needs a remote desktop Login of the virtual machine you created using EC2 instance.
2. Need to create a webpage using Webserver IIS to demonstrate the importance of cloud.
3. Apache Web Server 2 must be installed on EC2 Machine (having Ubuntu 20.04).
4. SSD must be less than 30GB.
5. SSH secured login only from the IP provided by Mr. Hemant.

Ans: To create a remote desktop login of the virtual machine using EC2 instance:

- Launch an EC2 instance with a Windows AMI that supports RDP. You can find the list of Windows AMIs here.
- Create a key pair and download the private key file (.pem) to your local machine.
- Configure the security group for your instance to allow inbound RDP traffic from your IP address on port 3389. You can use the win_set_firewall tool to do this easily.
- Get the public IP address of your instance from the EC2 console or using the ec2-describe-instances command.
- Get the administrator password for your instance from the EC2 console or using the ec2-get-password command. You will need to provide the path to your private key file for decryption.
- Connect to your instance using an RDP client such as Microsoft Remote Desktop. Enter the public IP address, administrator username and password when prompted.

To create a webpage using Webserver IIS, you need to follow these steps:

- Install IIS on your Windows instance using the win_install_iis tool or following the instructions here.
- Create a folder for your website files in your instance, such as C:\inetpub\wwwroot\mysite.
- Copy your website files to the folder using FTP, SCP, or any other method you prefer.
- Configure IIS to host your website using the win_configure_iis tool or following the instructions here. You will need to specify the site name, physical path, binding type, IP address, port, and host name for your website.
- Test your website by accessing it from a web browser using the public IP address or host name of your instance.

To install Apache Web Server 2 on Ubuntu 20.04, you need to follow these steps:

- Connect to your Ubuntu instance using SSH and your private key file.
- Update the package index and install Apache using the following commands:
  - sudo apt update
  - sudo apt install apache2
- Verify that Apache is running and enabled by using the following commands:
  - sudo systemctl status apache2
  - sudo systemctl is-enabled apache2

- Configure the firewall to allow HTTP traffic on port 80 using the ufw tool or following the instructions here.
- Test your Apache web server by accessing it from a web browser using the public IP address of your instance. You should see the default Ubuntu Apache page.

To choose an SSD size for your EC2 instance, you need to consider the following factors:

- The type and size of your EC2 instance. Some instances offer NVMe SSD instance store volumes that are physically attached to the host server and provide high performance and low latency. However, these volumes are ephemeral and do not persist after the instance is stopped or terminated. Other instances support EBS SSD volumes that are network-attached and provide persistent and durable storage. You can find more information about the different types of SSD volumes here.
- The storage capacity and performance requirements of your application. You should estimate how much disk space and IOPS (input/output operations per second) you need for your application and choose an SSD size that meets or exceeds those requirements. You can use CloudWatch metrics to monitor the disk usage and performance of your SSD volumes.
- The cost of SSD storage. SSD storage is more expensive than HDD storage, so you should choose an SSD size that fits your budget and provides enough space for future growth. You can use the AWS Pricing Calculator to estimate the cost of SSD storage for different types of instances and volumes.

To restrict SSH access to a specific IP address for a user in Ubuntu, you need to follow these steps:

- Connect to your Ubuntu instance using SSH and your private key file.
- Edit the SSH daemon configuration file (/etc/ssh/sshd_config) using a text editor such as nano or vi. You can use the sudo nano /etc/ssh/sshd_config command to open the file with nano.
- Uncomment the AllowUsers line and specify the username and IP address from which the user is allowed to connect. For example, if you want to allow user1 to connect only from 192.168.0.10, you can add this line:
  - AllowUsers user1@192.168.0.10
- Save and close the file.
- Restart the SSH daemon for the changes to take effect by using this command:
  - sudo systemctl restart sshd

P4: Assume, you are technical advisor in your organization. The organization's vision is to provide use the cloud services in AWS. You are directed to give Roles, authentication and authorizations to employees using a particular service of cloud.

Ans: As a technical advisor, I can use AWS Identity and Access Management (IAM) to manage roles, authentication and authorization for my employees using cloud services. IAM is a service that helps to control access to AWS resources by creating and managing users, groups, roles, and policies.

Here are some steps you can follow to use IAM effectively:

- Create IAM users for each employee who needs access to AWS resources. Each user can be assigned a unique set of credentials, such as a password and an access key, to authenticate with AWS. Multi-factor authentication (MFA) for additional security can also be enabled.

- Organize your IAM users into groups based on their job functions or responsibilities. For example, you can create a group for developers, another group for managers, and so on. You can then attach policies to each group that define the permissions for the group members. This way, you can manage permissions for multiple users at once, instead of individually.

- Use IAM roles to delegate temporary access to AWS resources for specific tasks or scenarios. For example, you can create a role that allows an employee to access a specific S3 bucket for a limited time, or a role that allows an external partner to access your AWS resources. You can also use roles to grant permissions to applications or services that run on AWS, such as EC2 instances or Lambda functions.

- Use policies to define the actions and resources that are allowed or denied for your IAM identities (users, groups and roles). Policies are JSON documents that specify the effect (allow or deny), the action (such as s3:PutObject), the resource (such as arn:aws:s3:::example-bucket/*), and optionally the condition (such as the IP address or the time of the request).

P5: Demonstration Elastic Load balancing using ECS in AWS

Tasks

Step 1: Configure a target group

Step 2: Register targets

Step 3: Configure a load balancer and a listener

Step 4: Test the load balancer

Step 1: Configure a Target Group

1. Login to AWS Console: Log in to your AWS Management Console.
2. Go to EC2 Dashboard: Navigate to the EC2 Dashboard and click on "Target Groups" in the left sidebar.
3. Create a Target Group: Click on the "Create target group" button. Configure the target group with the desired settings. For ECS, ensure the target type is set to "IP".

```
$ targetGroup=$(awslocal elbv2 create-target-group --name example-target-group \
        --protocol HTTP --target-type ip --port 80 --vpc-id $vpc_id \
        | jq -r '.TargetGroups[].TargetGroupArn')
```

Step 2: Register Targets

1. Select Your Target Group: Select the target group you created and go to the "Targets" tab.
2. Register ECS Tasks as Targets: Click on the "Edit" button, then select your ECS cluster and service. Register the appropriate ECS tasks as targets.

```
$ awslocal elbv2 register-targets –targets Id=127.0.0.1,Port=5678,AvailabilityZone=all \
  --target-group-arn $targetGroup
```

Step 3: Configure a Load Balancer and a Listener

1. Create Load Balancer: In the EC2 Dashboard, click on "Load Balancers" in the left sidebar. Create a new load balancer, choosing the appropriate VPC and subnets.
2. Configure Listener: Configure a listener for the load balancer. For example, if you're setting up a web application, configure a listener on port 80 (HTTP) or 443 (HTTPS). Associate the listener with the target group you created.

```
$ listenerArn=$(awslocal elbv2 create-listener \
     --default-actions
'{"Type":"forward","TargetGroupArn":"'$targetGroup'","ForwardConfig":{"TargetGroups":[{"Target
GroupArn":"'$targetGroup'","Weight":11}]}}' \
     --load-balancer-arn $loadBalancer | jq -r '.Listeners[]|.ListenerArn')
```

```
$ listenerRule=$(awslocal elbv2 create-rule \
     --conditions Field=path-pattern,Values=/ \
     --priority 1 \
     --actions
'{"Type":"forward","TargetGroupArn":"'"$targetGroup"'","ForwardConfig":{"TargetGroups":[{"TargetGroupArn":"'"$targetGroup"'","Weight":11}]}}' \
     --listener-arn $listenerArn \
 | jq -r '.Rules[].RuleArn')
```

Step 4: Test the Load Balancer

1. Access Load Balancer DNS: Once the load balancer is active, note down its DNS name.
2. Test the Load Balancer: Use a web browser or a tool like curl to access the DNS name of your load balancer. Requests will be distributed across the ECS tasks registered in the target group.

```
$ listenerRule=$(awslocal elbv2 create-rule \
        --conditions Field=path-pattern,Values=/ \
        --priority 1 \
        --actions
'{"Type":"forward","TargetGroupArn":"'"$targetGroup"'","ForwardConfig":{"TargetGroups":[{"TargetGroupArn":"'"$targetGroup"'","Weight":11}]}}' \
        --listener-arn $listenerArn \
    | jq -r '.Rules[].RuleArn')
```

OUTPUT:

```
{
  "host": {
    "hostname": "example-lb.elb.localhost.localstack.cloud",
    "ip": "::ffff:172.17.0.1",
    "ips": []
  },
  "http": {
    "method": "GET",
    "baseUrl": "",
    "originalUrl": "/",
    "protocol": "http"
  },
  "request": {
    "params": {
      "0": "/"
    },
    "query": {},
    "cookies": {},
    "body": {},
    "headers": {
      "accept-encoding": "identity",
      "host": "example-lb.elb.localhost.localstack.cloud:4566",
      "user-agent": "curl/7.88.1",
      "accept": "*/*"
    }
  },
  "environment": {
    "PATH": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "HOSTNAME": "bee08b83d633",
    "TERM": "xterm",
    "NODE_VERSION": "18.17.1",
    "YARN_VERSION": "1.22.19",
    "HOME": "/root"
  }
}
```