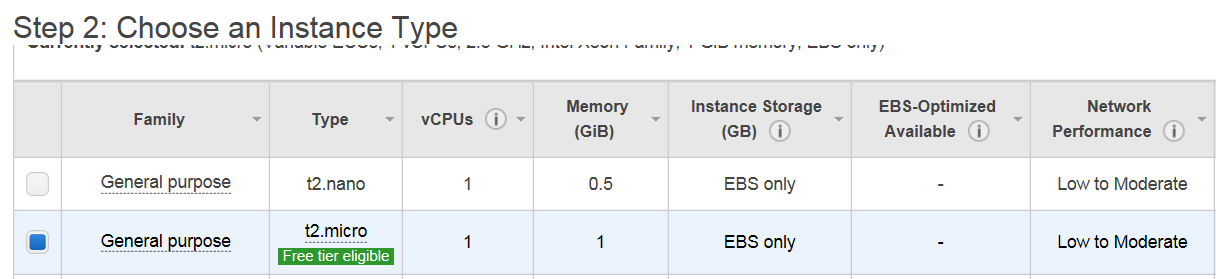
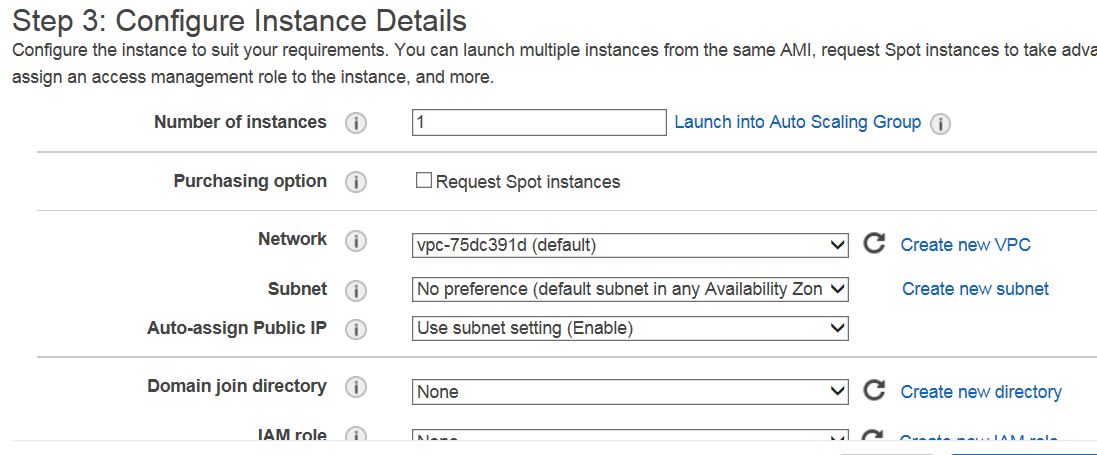
Instance of EC2:

EC2 dashboard. Launch instance. Windows server



Next



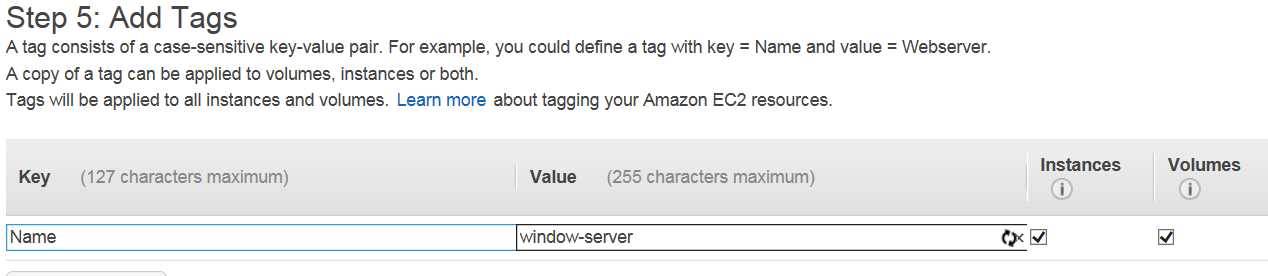
Next

With windows, min has to be 30gb.

Next

Tags:

Value : window-server



Next:

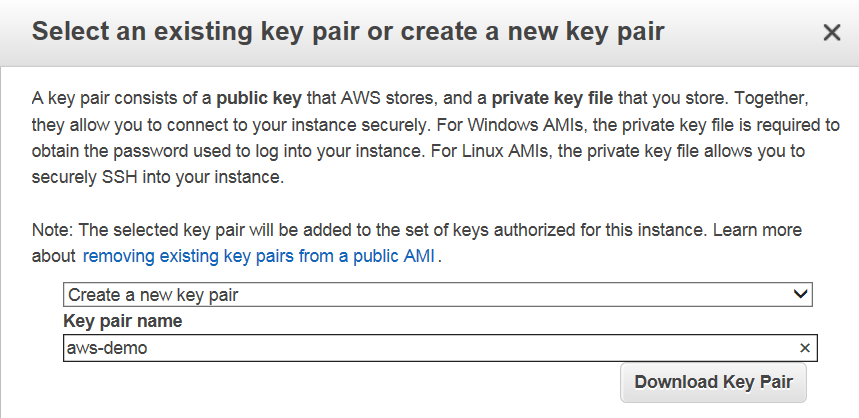
Security Group:

To control inbound and outbound traffic. Launch.

Public ke kept with aws. Private key we need to save.

Create a new ke pair

Aws-demo

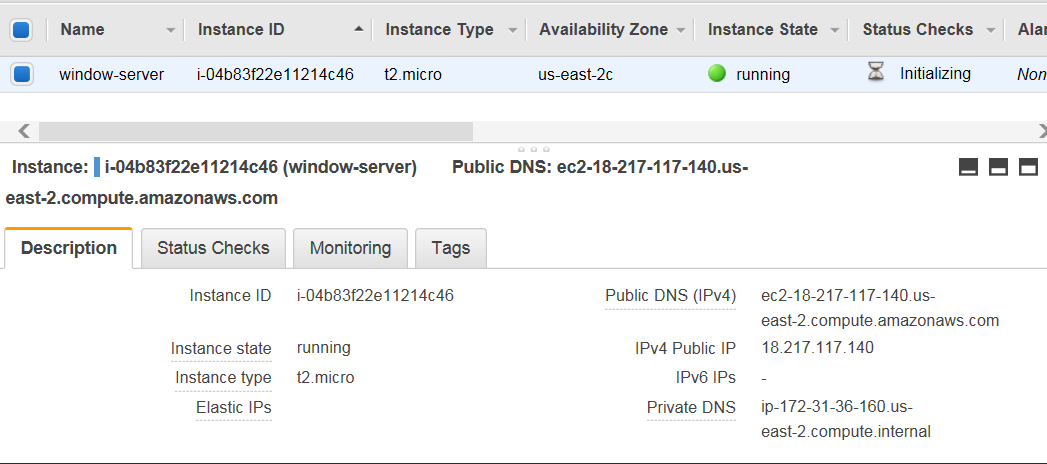


Download key pair

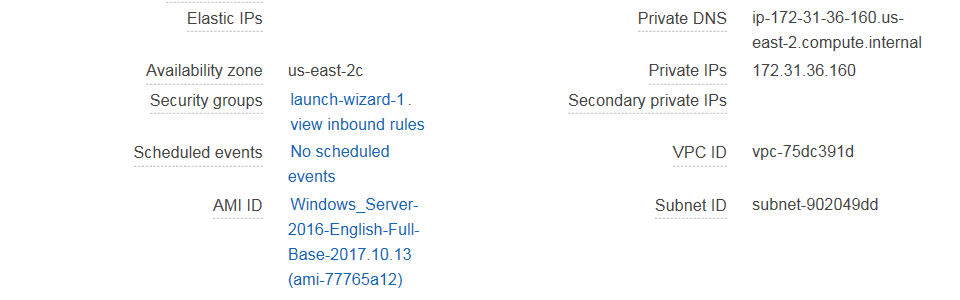
Save the private file

Launch the instance.

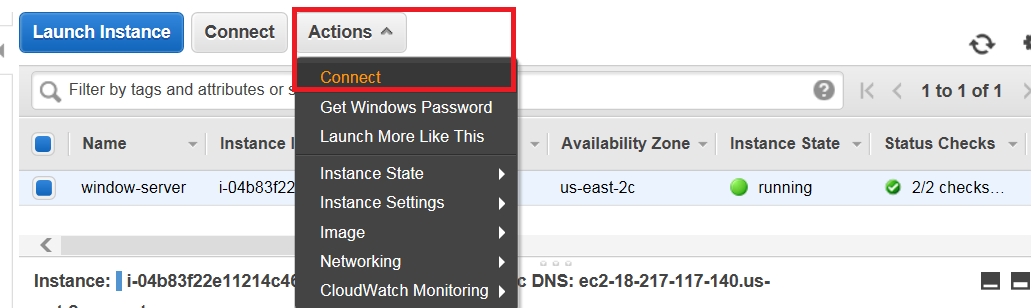
Can now see the instance in the console



All the properties listed below.

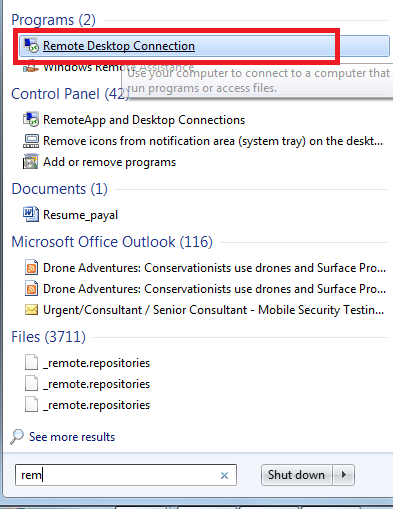


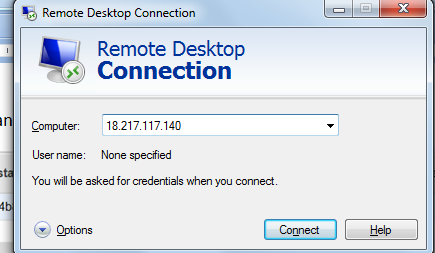
Public IP would be used to connect to the instance. Connect using remote desktop connection. Username for windows is by-default : “Administrator” and password: upload the file. It will decrypt the password using private key and public ke file.



After instance is running, finished Initialization, click on Actions -> Connect. You’ll get the decrypted password from there by uploading private file.

Launch the remote desktop connection:





Enter IPV4 public IP

Click connect

Enter username: Administrator

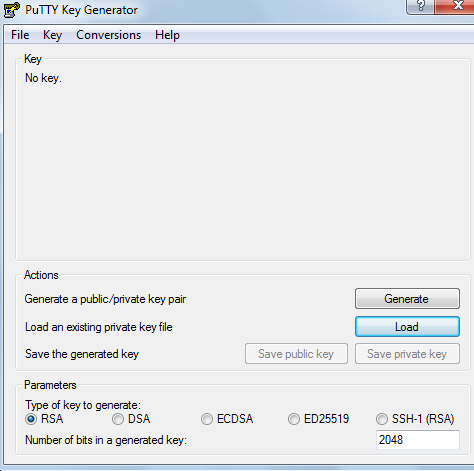
Password: decrypted password

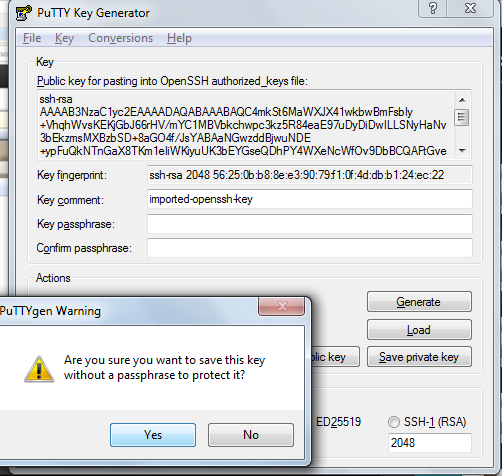
Connect

Server Instance would be launched.

Create ubuntu server. Connect using putty.

Launch putty gen. Load private key .

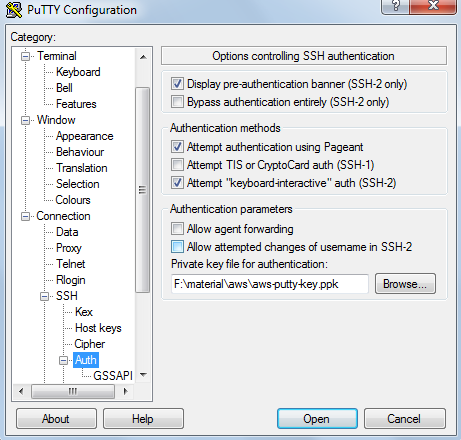




Save private key. So now pem file converted to ppk.

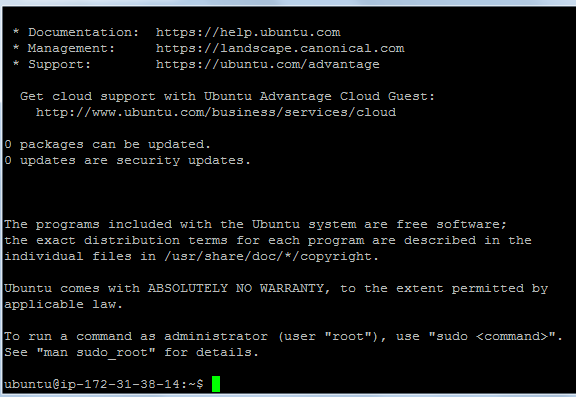
Open putty. Enter id.

SSH-> auth-> upload ppk file. Click Open, It will connect to ip



Username: ubuntu

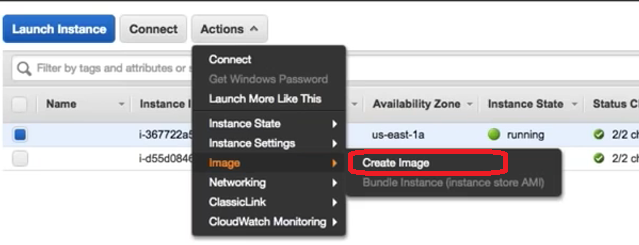
Hit enter. Successfully connected.

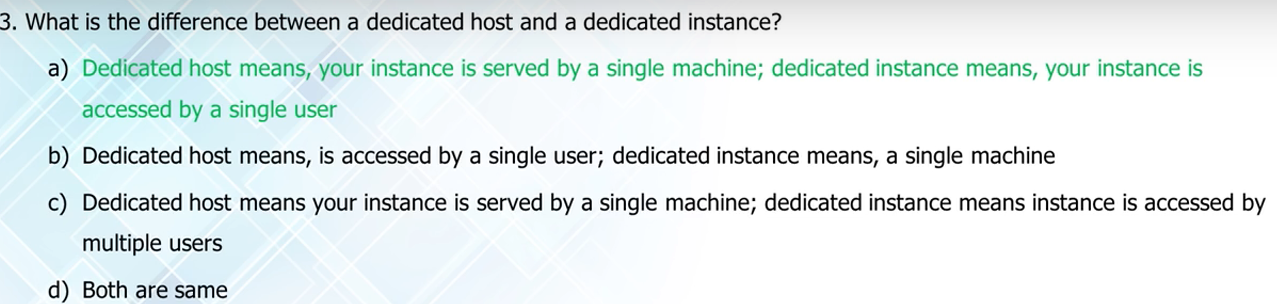


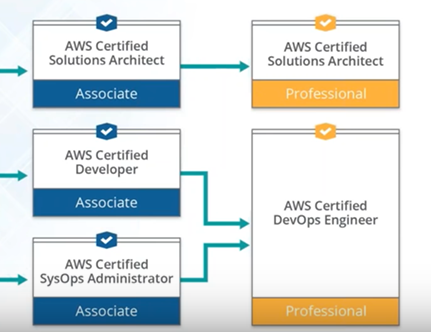
For linux server:



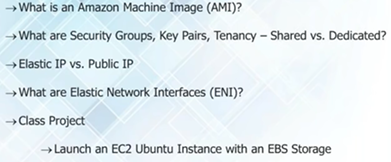
Once server is created, we can install softwares on it and then create as many images of the server as we like.

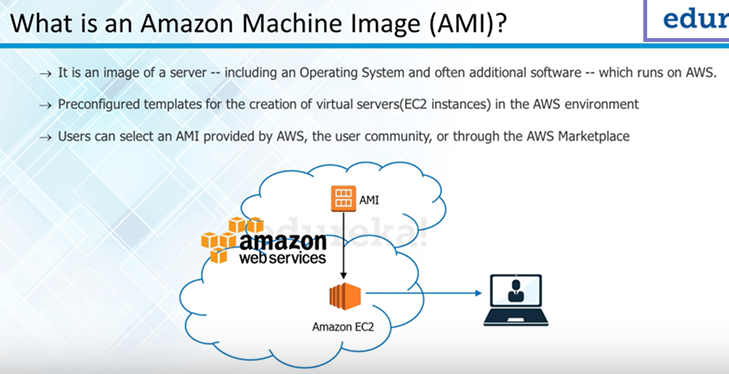


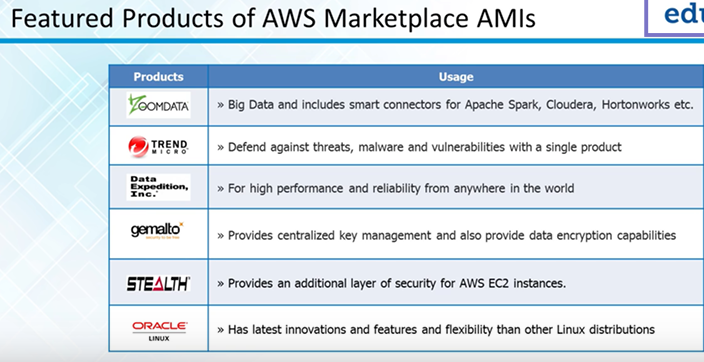


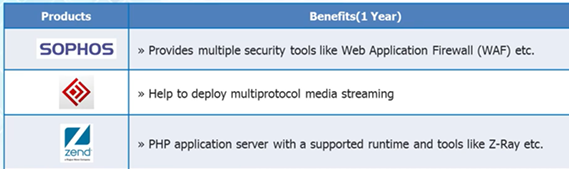


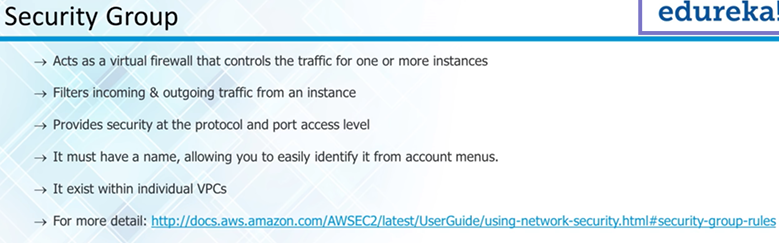
Amazon EC2:

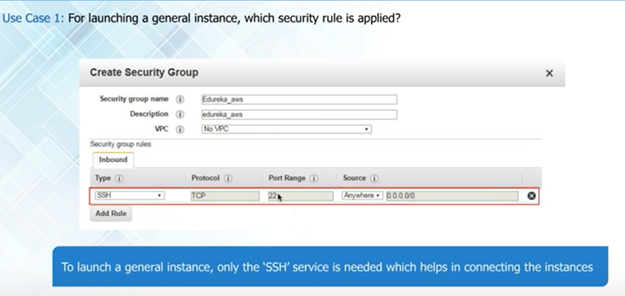




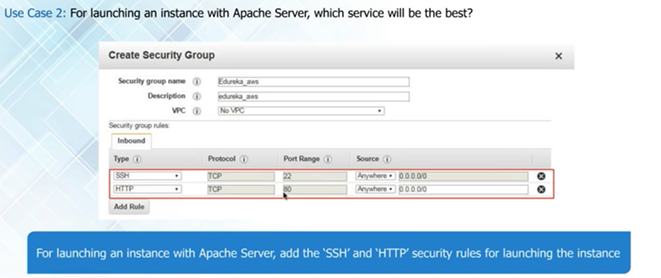






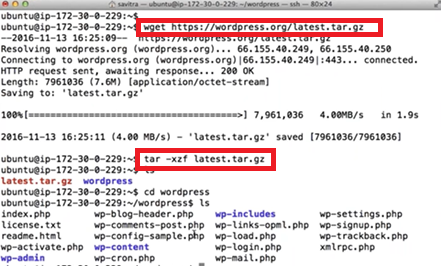


Anyone can connect to this server as source is 0.0.0.0



Can add other rules. HTTP port can be enabled for all. Accessible to all

**Deploying wordpress on ubuntu server**



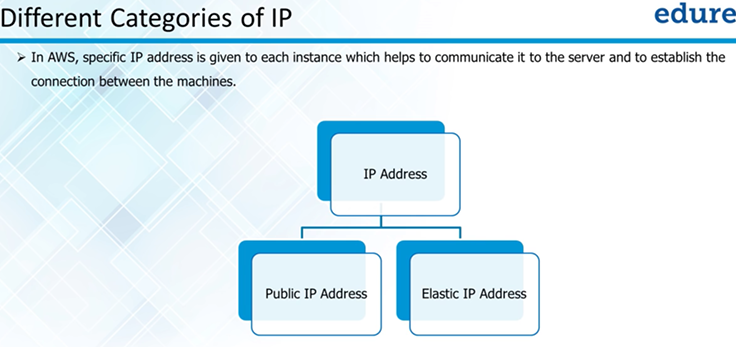
**Install mysql**

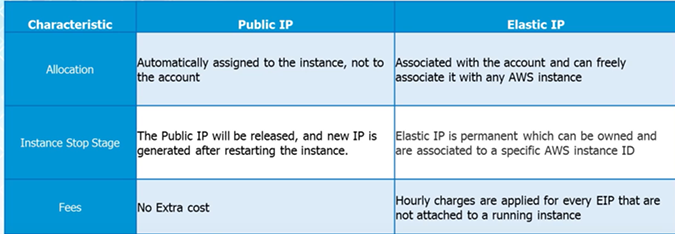
1. Sudo apt-get update

Get current version of the softwares

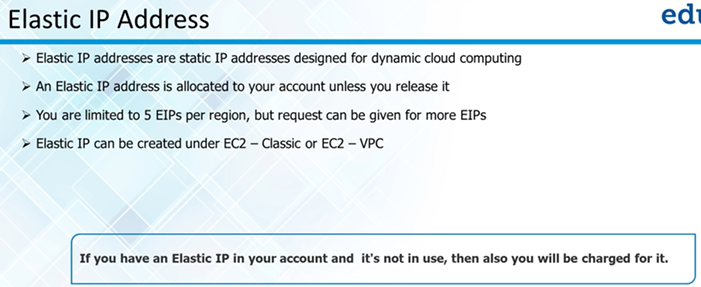
1. 

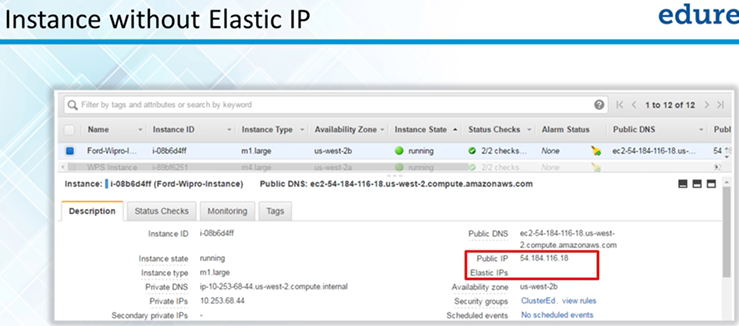
Install php5 and mysql server ,apache2 server





Domain name cannot point to public ip address as they are volatile.

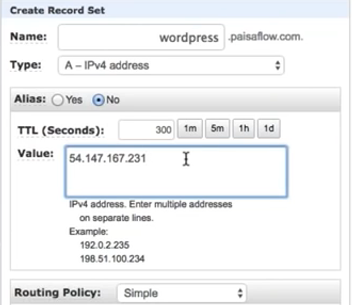




**Creating Elatic IP:**

Route53:

Can buy a domain. If we have a domain, we can manage it through route53 using “hosted zone”. Create hosted zone:



So we have public ip domain pointing to current ip.

host a website: buy servers, peak traffic time: buy more server. Monitor and mantain server.

Set up is expensive. Troubleshooting problem expensive. Server idle most of the time.

cloud computing: put data on cloud servers. Scalability. Cloud provider manages the server.

aws: secure cloud service platform, offering compute power, db storage, other functionalities

to help business scale and grow.

domains in which aws offers services: compute (EC2: raw server to host website, like a new pc,

install os and other softwares), migration (transfer data physically to aws or transfer back

from aws), security and identity, compliance (IAM: to authenticate users and give rights) ,

storage (S3: simple storage server: file system ),networking and content delivery(Route 53:

redirect traffic to instances/ servers hosting website), messaging(SES: simple email

service:send emails in bulk to customer, handles reply), database( cannot include executable

file eg image , path saved in db RDS: relational database service: manages database for us i.e

automatically update db engine or do automatic commit), management tools

Content delivery: if user far from server which he's trying to access To reduce response time.

aws services: aws compute domain:

1) EC2 (elastic compute cloud): base , RAW SERVER cab be configured to be web server or

anything. size can be reconfigured according to no. of servers being used.

2) Lambda : advanced version of EC2. Cannot be used to host but to execute our background

tasks. Image compressed and stored in file system. Applying filters.

3) Elastic Beanstalk: advanced version of EC2. Used to host an application. Automated form of

EC2. We don't have to set-up the environment. PHP website to be hosted, php env to be set-up.

Just select type of env. Automatic env set-up. Deploy website.Has limited number of

environments. If env not listed, should use EC2.

4) Elastic load balancer: distribute work load on number of instances. 4-5 servers running. all

traffic on 1 server. Not efficient. Work-load distributed

5) Autoscaling: to scale up and down automatically without human intervention. If combined cpu

usage beyond 70%, host a new server. Load balancer will automatically bakance load. Can apply

metrics. Can scale up or scale down.

Ews comsole. sign in.EC2: launch instance: ami: operating system: windows server . t2 micro.

number : 1, network: vpc-c9ae08ae

subnet: subnet-cb9056ac| us-west-2b

next

30gb min

next

name windows-server

security group ( to control inbound outbound traffic)

RDP TCP 3389 Custom

Launch.

public key (with aws)| private key (can download)

name: aws-demo

download key pair. (private file)

EC2 instance launched. description: ip address, username: Administrator, password: upload

private file.

actions: connect.decrypt password: save password

remote desktop connection:

Storage domain:

1) S3: file system, object oriented. All files uploaded are treated as objects. Objects stored

in a bucket. Root folder is like a bucket. It can have folders inside it.Objects will have

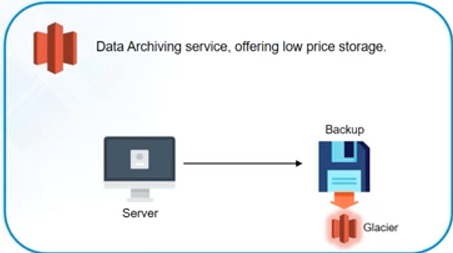
path.

2) Cloudfront: content delivery network: website far away from the user, caged to a server near

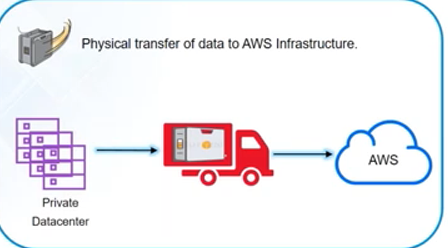
to user ( edge loction). Response time reduced.

3) eLASTIC BLOCK STORAGE: Like a hard drive. Os , softwares getting stored somewhere. 1 ec2 instance can be connected to many ebs volume but 1 ebs volume can be connected to only one EC2 instance.

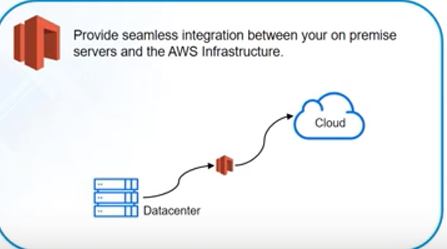
4) Glacier: data archiving service. Used to backup data. Use magnetic tapes to store data. Cheap. Data not frequently accessed. Store data not frequently accessed. Test records 6 months old.



5) snowball: physical device to transfer data from our data-center to aws infrastructure. When large amount of data (in picabytes) to be sent across. Data transferred in bulk



6) Storage gateway: seamless integration between your on premise server and aws infrastructure. Snapshot of dataservers stored in s3.



Will sit between data server and aws and will keep taking snapshot of data and storing it in s3. If a failure happens, will take the snapshot and restore the server with that snapshot configuration.

**AWS database domain:**

1)RDS : relational database management service. manages relational database (oracle, postgres, mysql..)for us.Installs security patches, updates db engines automatically.

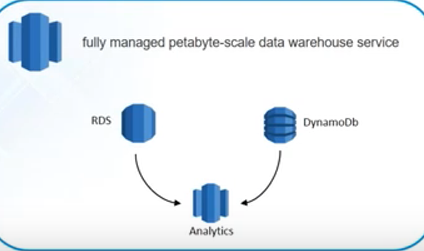
2) dynamodb: manages non-relational db for us. no-sql db. scaling db, security patches. . We don’t have to specify size. Automaticall scales if data increases.

3) amazon aurora: relational db , also managed by rds. Based on mysql. 5 times more faster than mysql

4) elasticcache: caching service. to manage and scale distributed cache-in env on cloud. same query being fired again, query result cachedin elastic cache.



5) red shift: data warehouse service. data feeded from rds, dynamodb, and does analysis of data.



**Launch rds service:**

database: mysql select

env: dev/test

next

db instance class: db 12 micro(smallest)

multi-availabiltiy zone env: no

storage type: general purpose ssd

dbinstnace identifier: my-demo

username: mysql

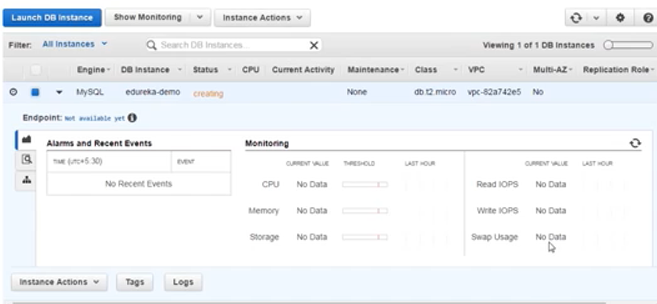
password: mysql

EC2 and rds have to be included in same vpc so that they can interact.

vpc: vpc-c9ae08ae

port: 3306

parameter group/version: 5.6



mysql instance on local server. link this instance with rds.

Using command line to link to this instance. First mysql should be installed on your system.

cmd: mysql/bin:

mysql -h <end-point> -P 3306 -u mysql -p

//-h host name -P port -p password -u username

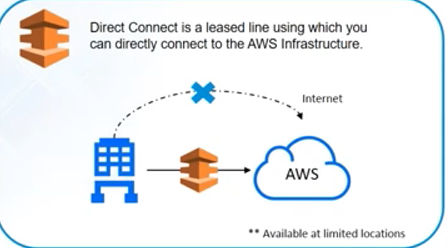
show databases;

can fire any query now.

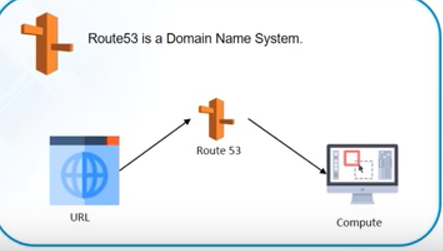
**Networking services:**

1) vpc: virtual private cloud. All aws resources in one vpc can interact with each other. security, can connect to private data center using vpn.

2) direct connect: replacement to Internet connection.leased line to directly connect to aws for data to flow



3) route 53: domain name server: request pointed from url to server on which website is running.

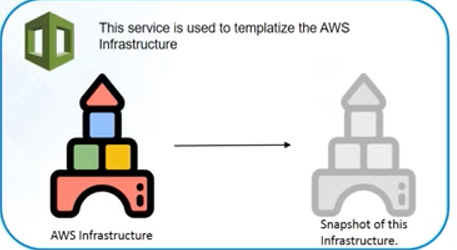


**Management domain:**

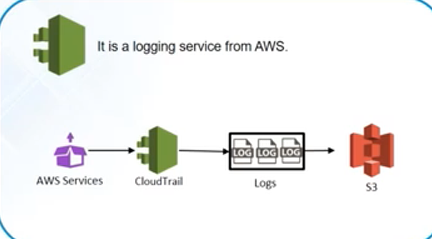
1. cloudwatch: monitoring tool. used to monitor all aws resources. if usage goes above 90%, send email. Can create alarms as per resources usage



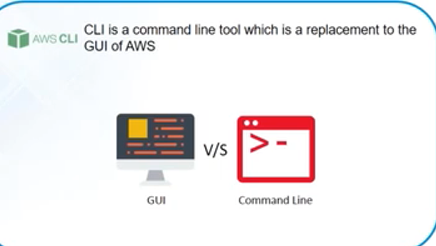
1. cloud formation: to templatize aws infrastrutcure. can take a snapshot and use it for other environments.



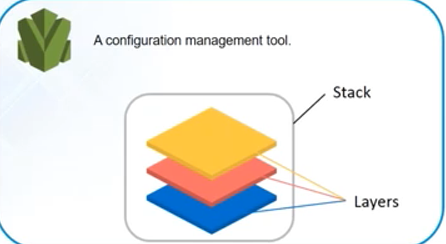
1. cloudtrail: logging service from aws. log all api request, reponse. logs saved in s3.



1. cli: command line interface, replacement to gui interface.

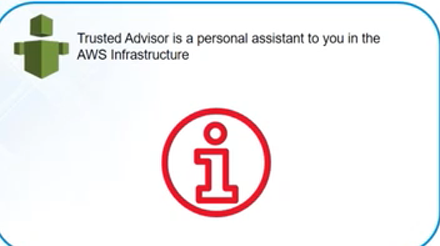


1. opsworks: configuration management tool. has stacks and layers ( aws services combined together, when combined together called stack). Either change can be done to each and every service or use opsworks to do change at stack level.



6) trusted advisor: like a personal assistant. advices on monthly expenditure. best practices. policies.

security services:



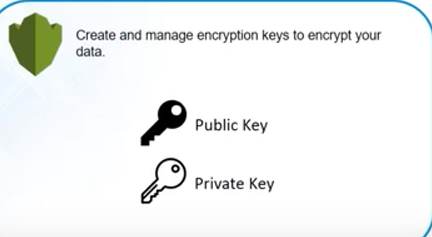
**Security Domain**

1. iam: identification and authorization management tool. user permissions to launch instance or just view them.



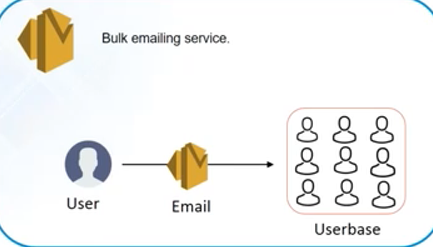
2) kms: key management service: any instance launched has public private key. public key with

aws. private key matched for authentication.

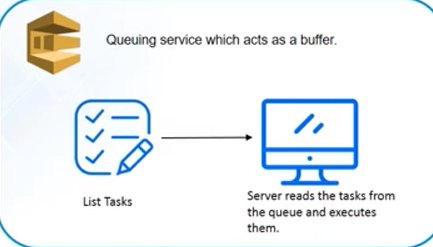


**Application services:**

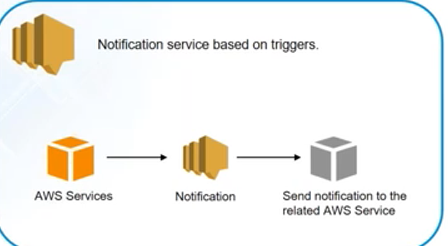
1. ses: simple email service : send email to large users. bulk emailing service, send automated replies.



1. sqs: simple queue service: acts as a buffer. image processing application. have to do 5 tasks. tasks in queue. multiple servers running. if 3 done by 1 server ,rest should know which are done and 2 left.



3) sns: simple notification service based on triggers. send notification to other aws services. image uploaded and email to be sent whenever any upload. sns sends notification to ses or sqs to send email or tasks to be added.



**AWS pricing:**

1)pay as you go: pay what you use.

2) pay less by using more. Hourly rate.

**save when you reserve:**

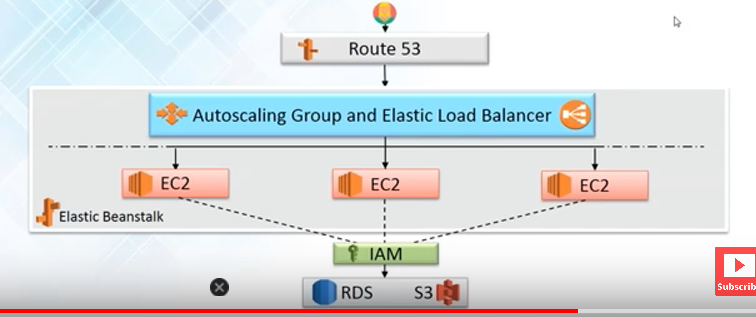
1) reserve your instances for time period. must be knowing how long we would be needing this instance. Eg: website would be there for long.videos may be deleted , so should not be reserved.

**Use case:**

hosting website on aws, used to upload images. highly available, autoscalable :



user: point to address: address to website : fileserver (images uploaded), db for path to images.

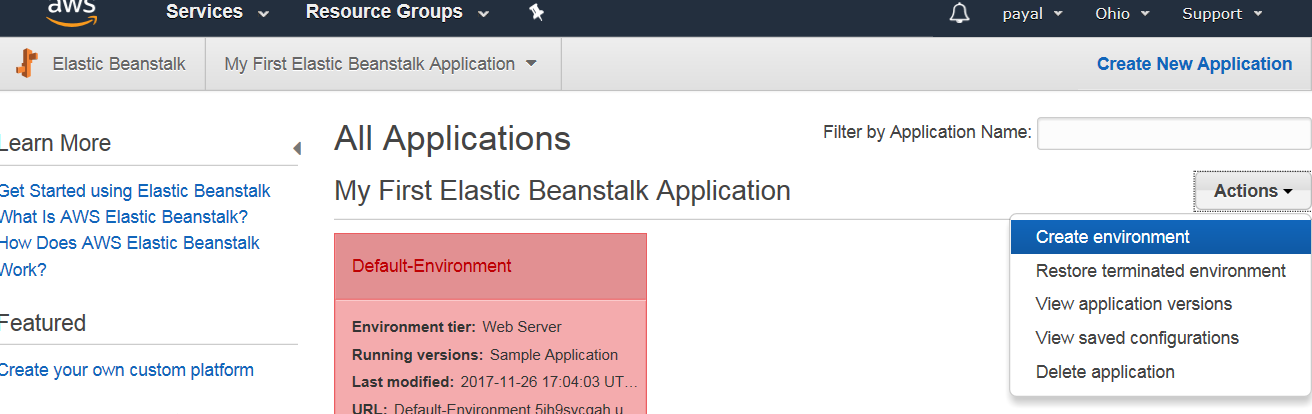


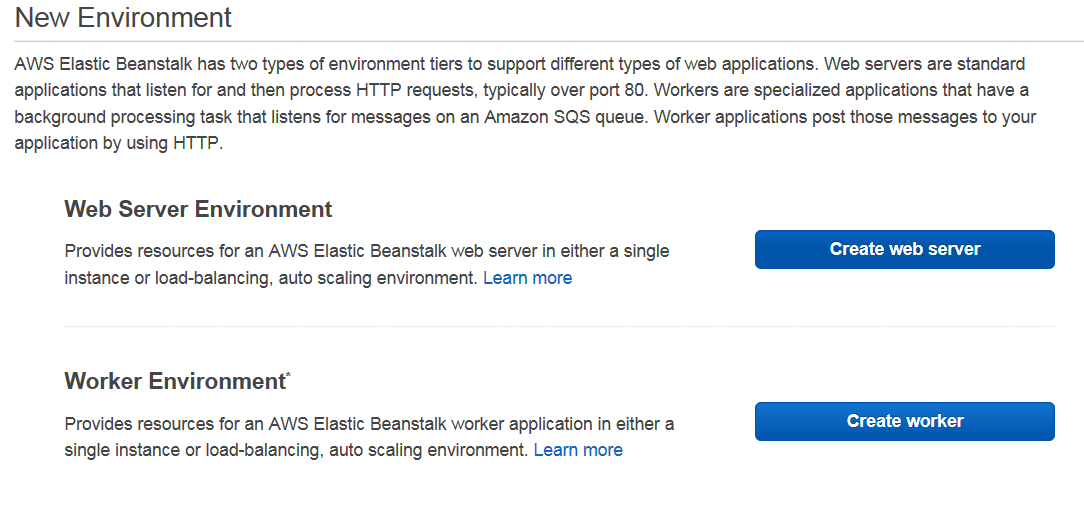
route53 ( to point to server hosting website), php application: elastic benastalk instead of

ec2. rds(db), s3 (file system), iam when using 3rd party tools for authentication.

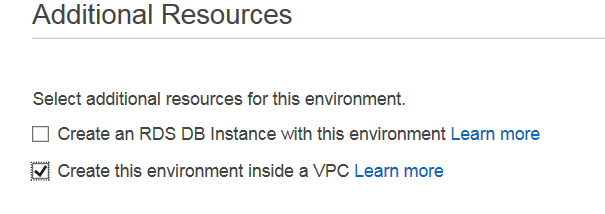
**Creating the above architecture and hosting website on aws**

Elastic Beanstalk: Create an environment.

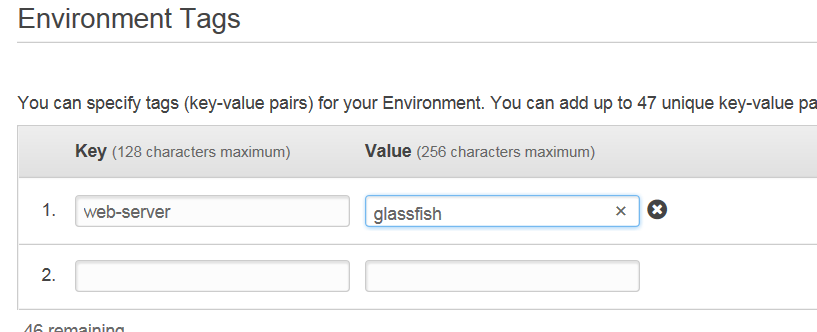




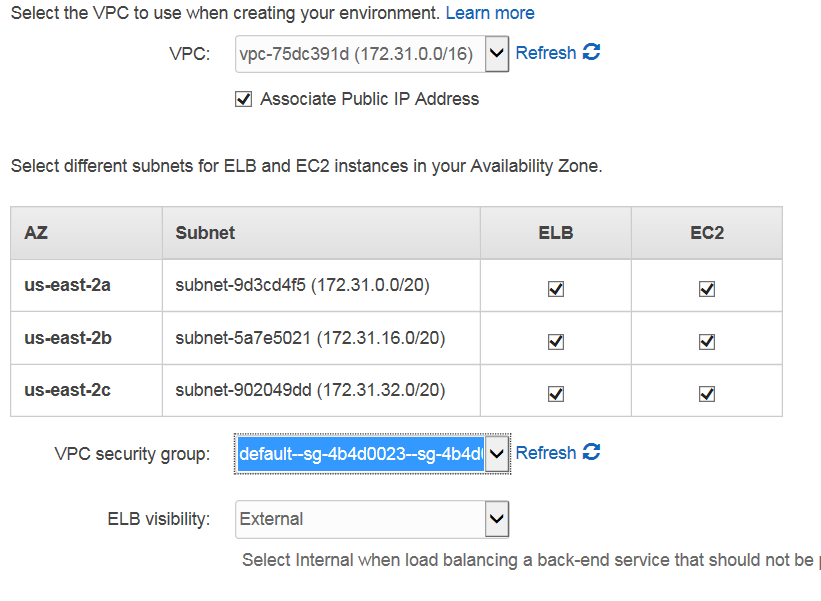
Create web server inside the vpc



Enter the name for the instance



Select same vpc as rds instance.



Next

**Create a backup of database instance and migrate the data**

Go to mysql bin directory. Create the dump and export it to the file with the name edureka.sql in the bin folder.



Connect to your db instance



So create a database here where data should be replicated

Create database edureka;

Use edureka;

Show tables; //empty

Use < to import in edureka database.

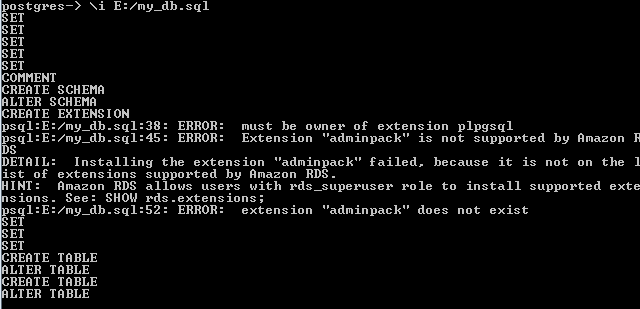


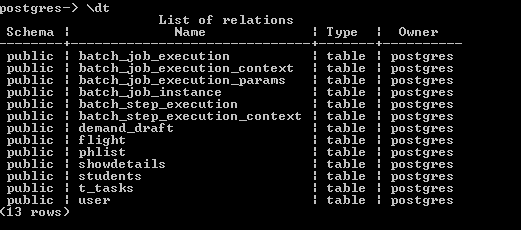
String edureka after –p is not the password but the database name. Password would be prompted later

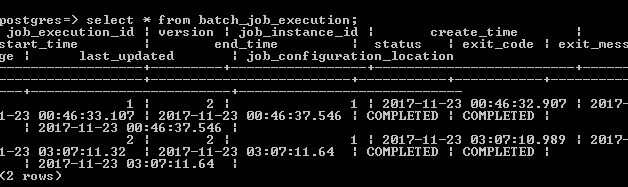
**Doing same for postgres**







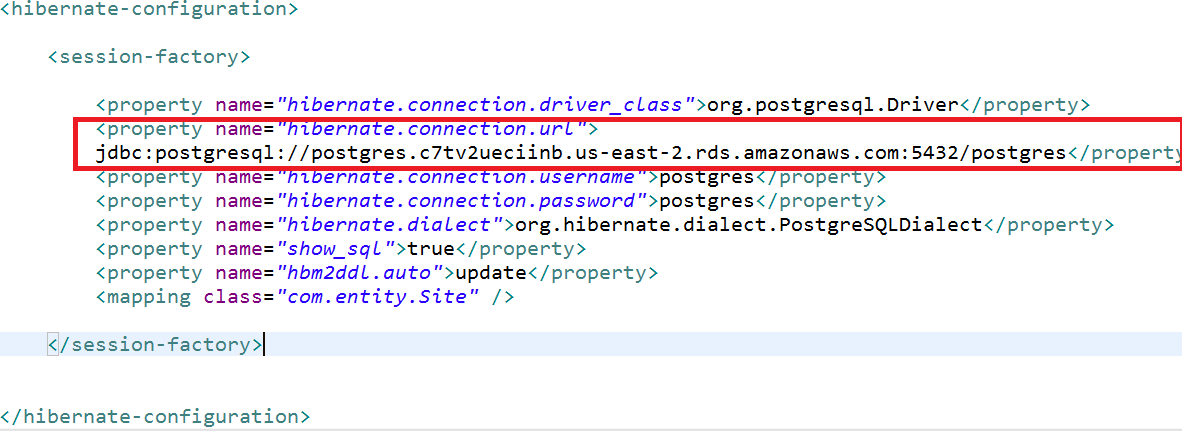




Import successful.

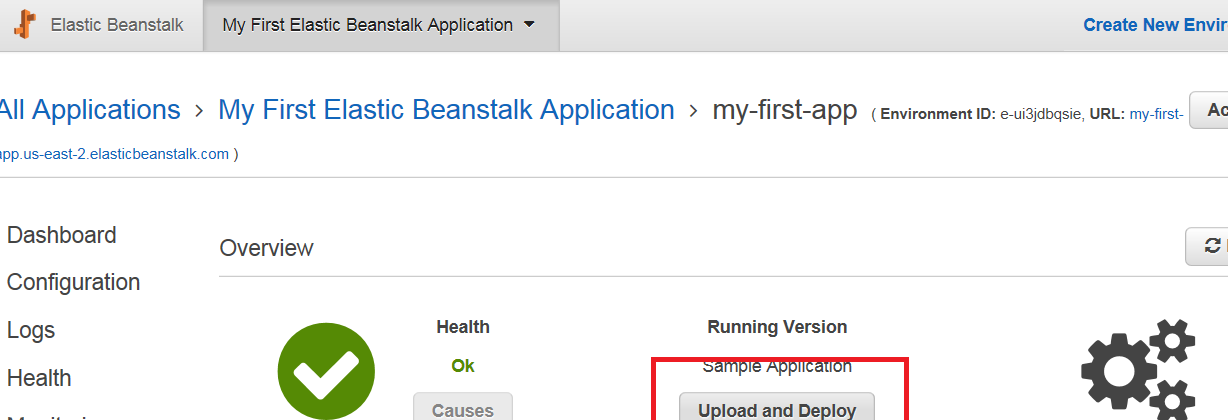
In the application, change the host name to be the endpoint url. Instead of localhost, it should be the endpoint url. So the application code would now be able to interact with the RDS.

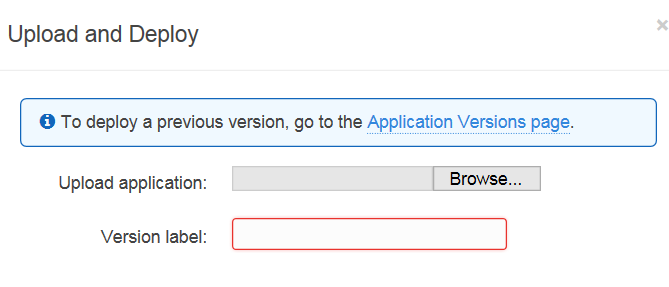
**Sample change to hibernate configuration**

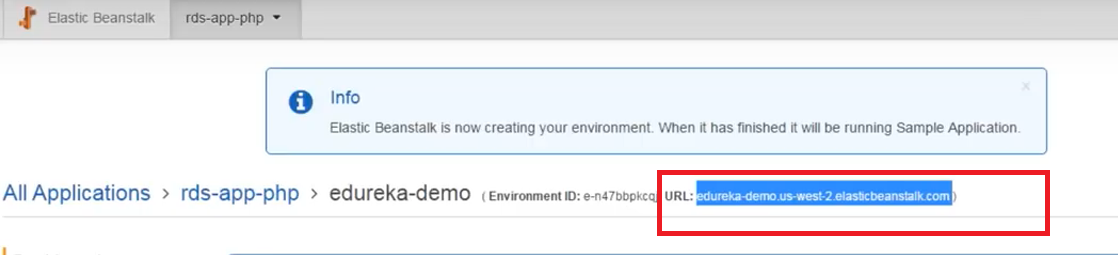


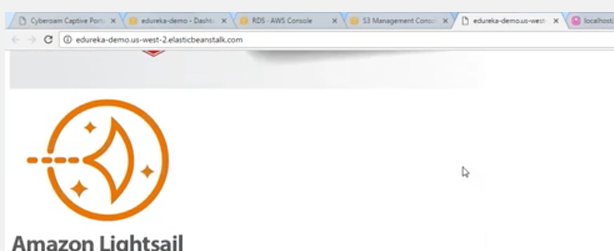
After this code interacting with rds.

**Upload the application to AWS**

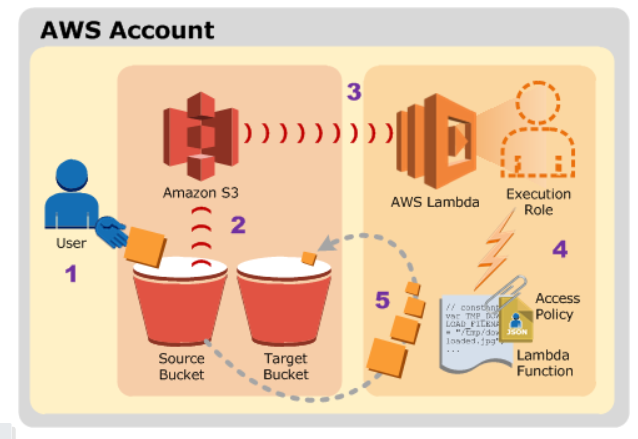
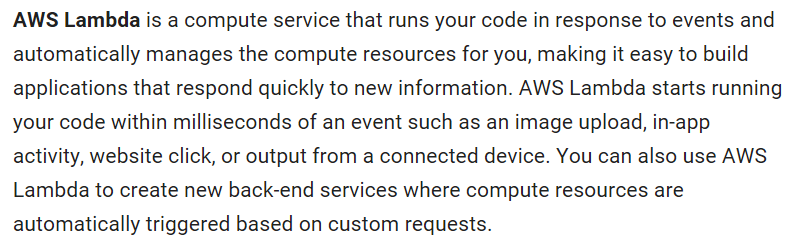
****

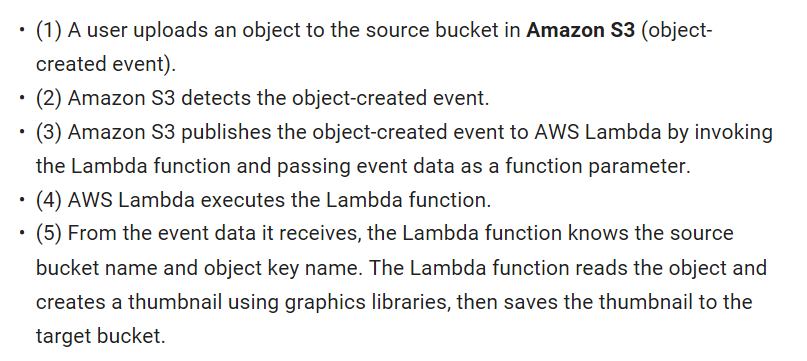
****

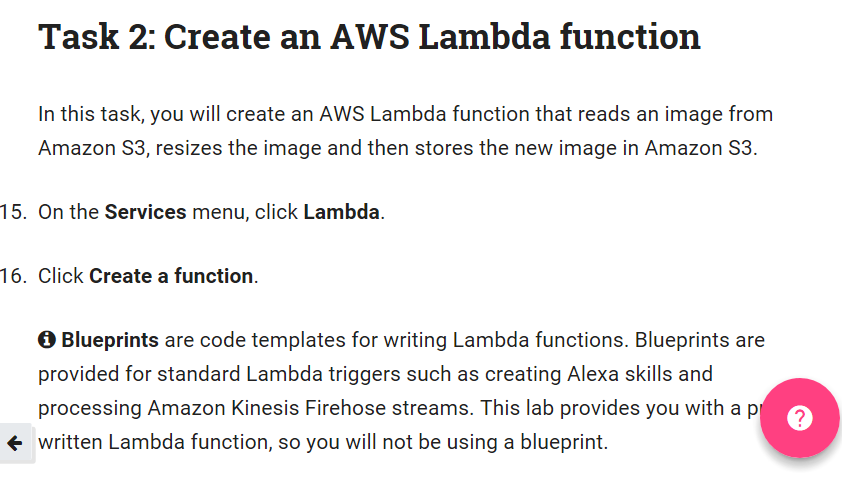
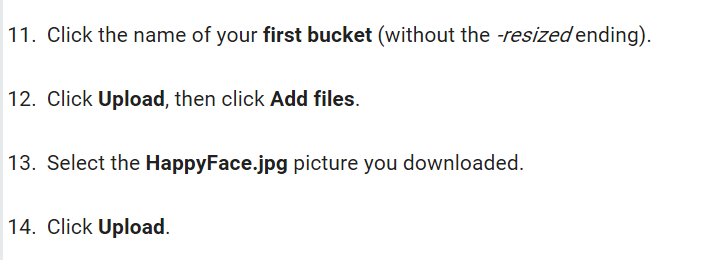
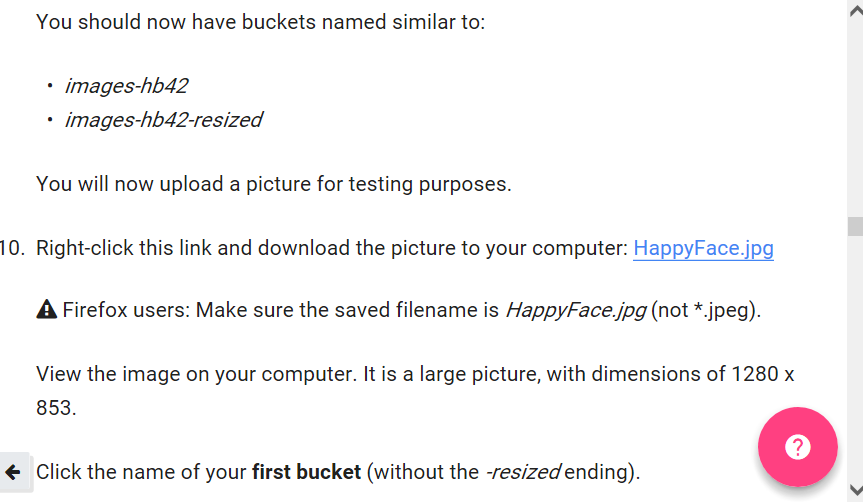
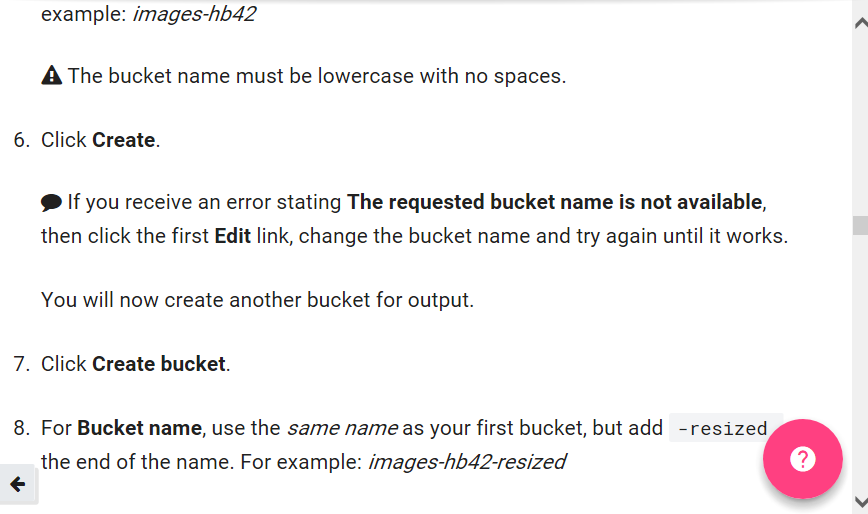
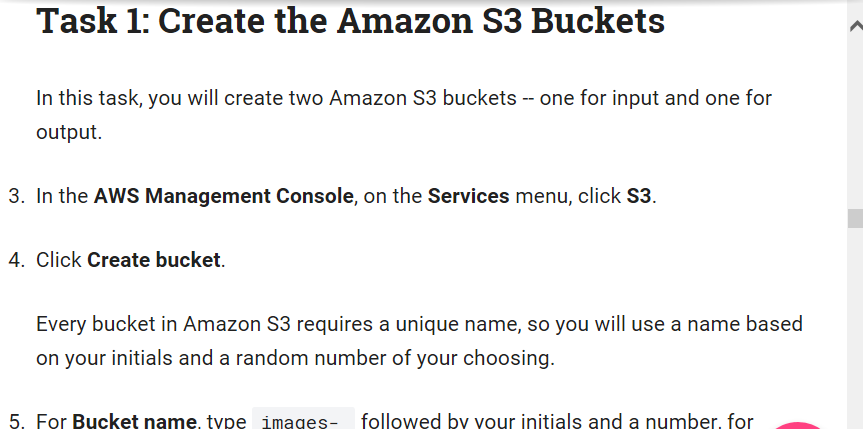
Choose the application. After upload will see the [url:](url:)

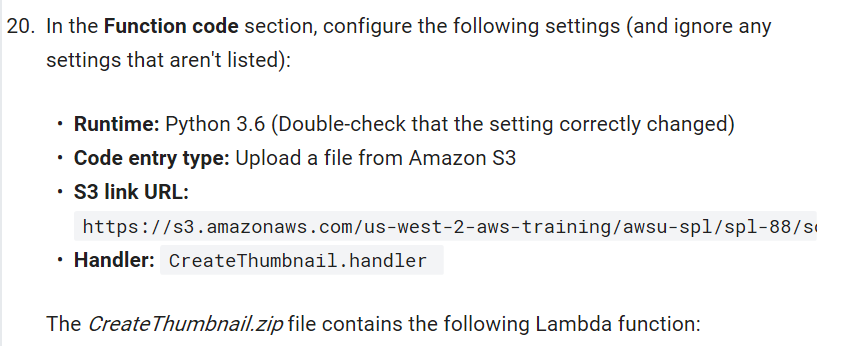
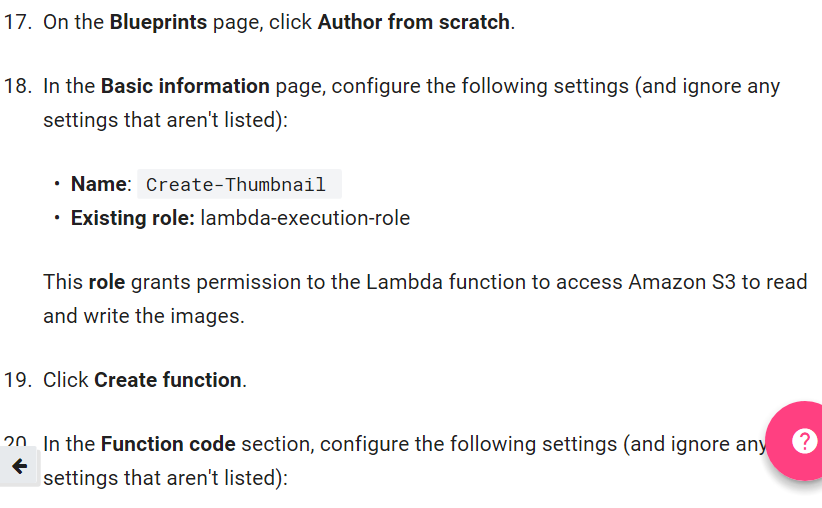
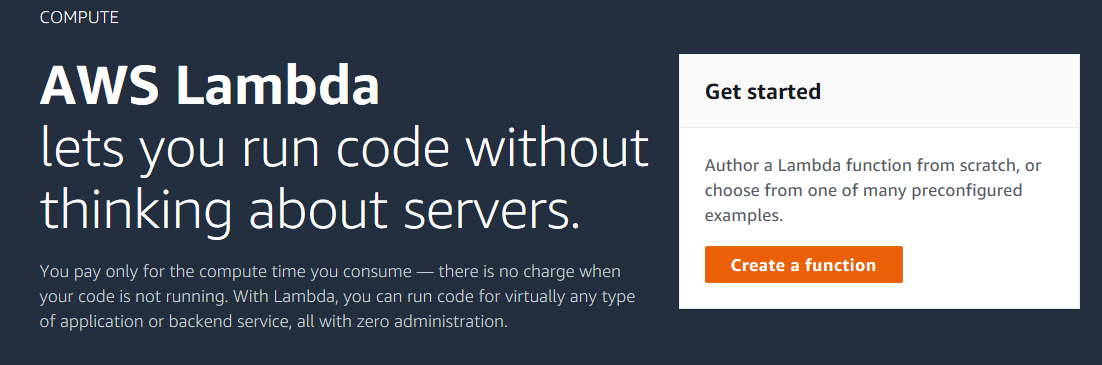
Site now accessible at that url: 

**AWS self paced labs: https://aws.amazon.com/training/self-paced-labs/**

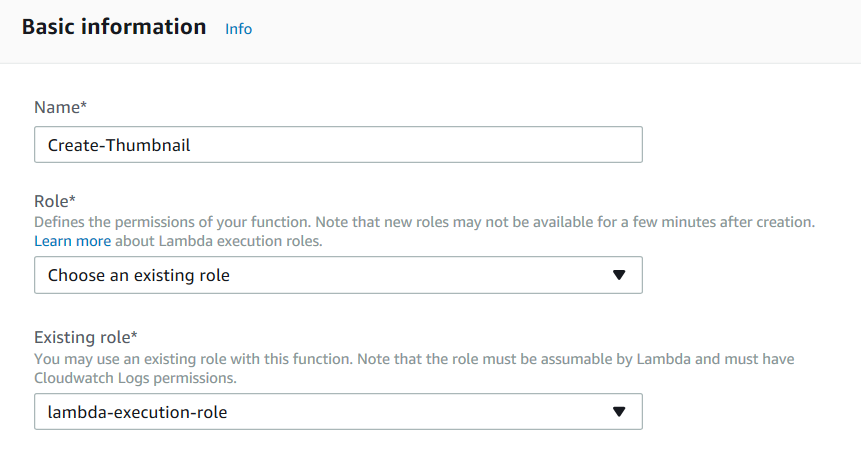
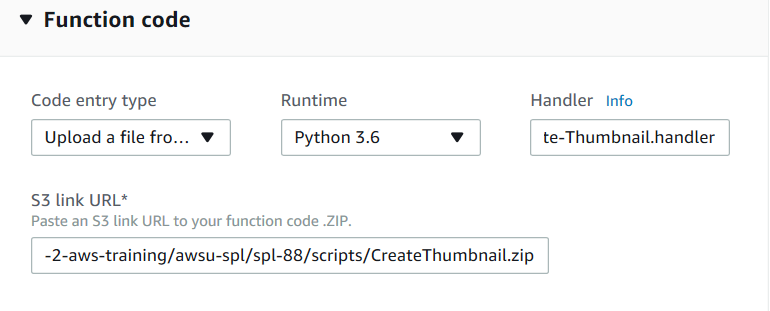


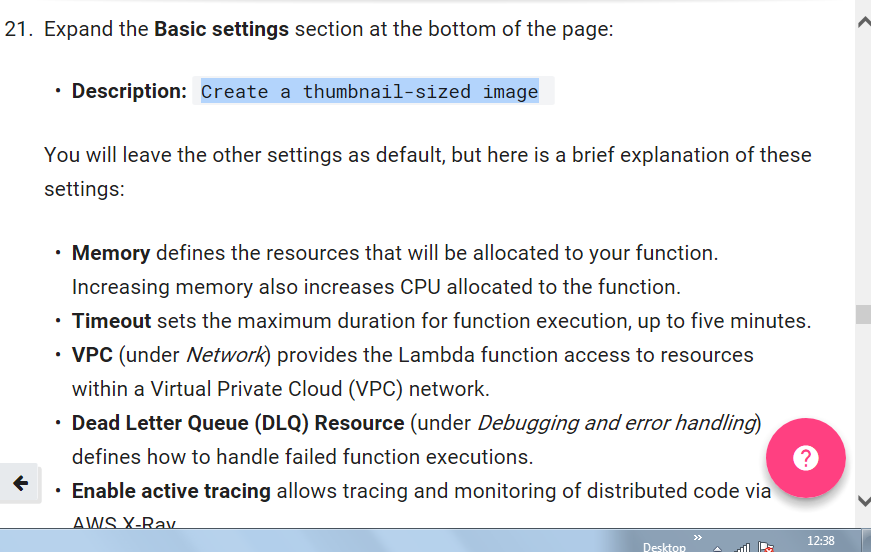


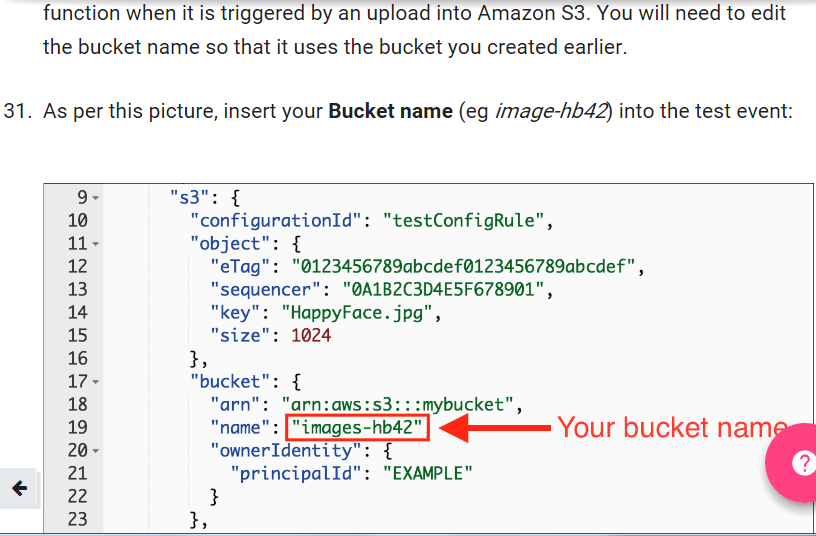
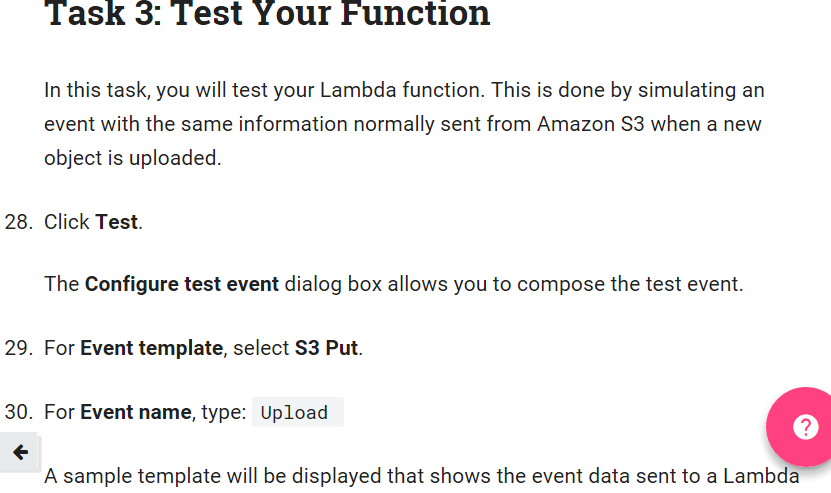
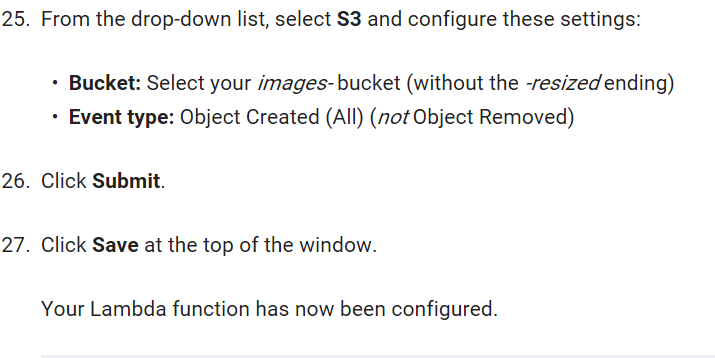
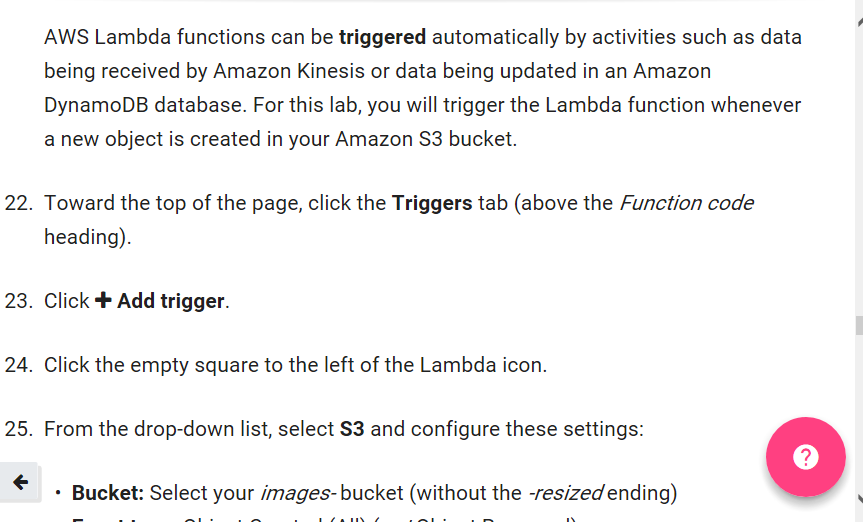


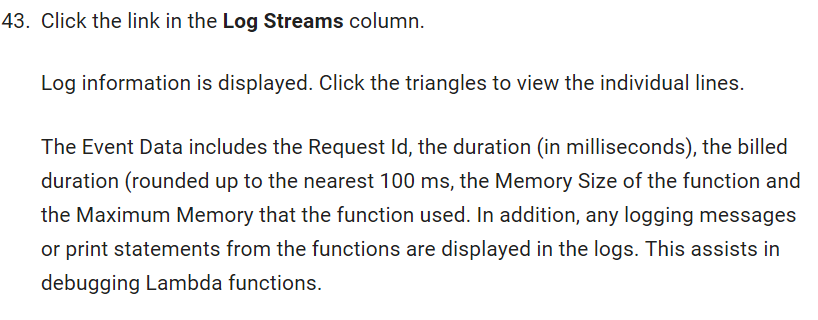
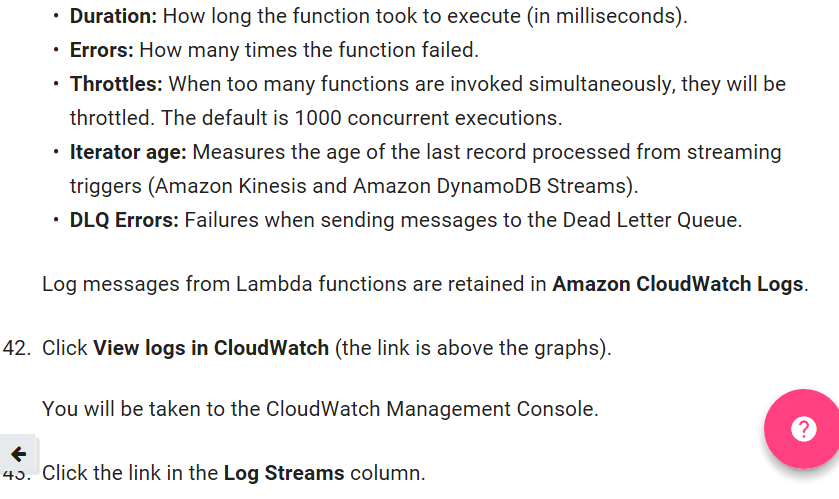
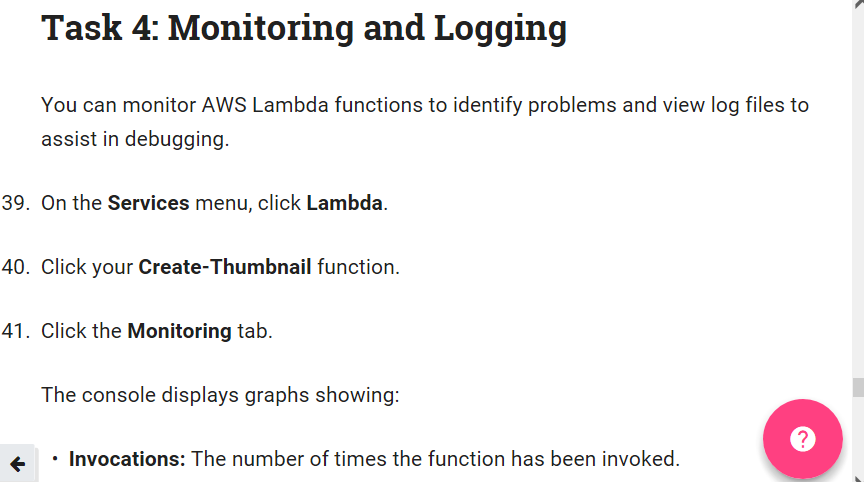
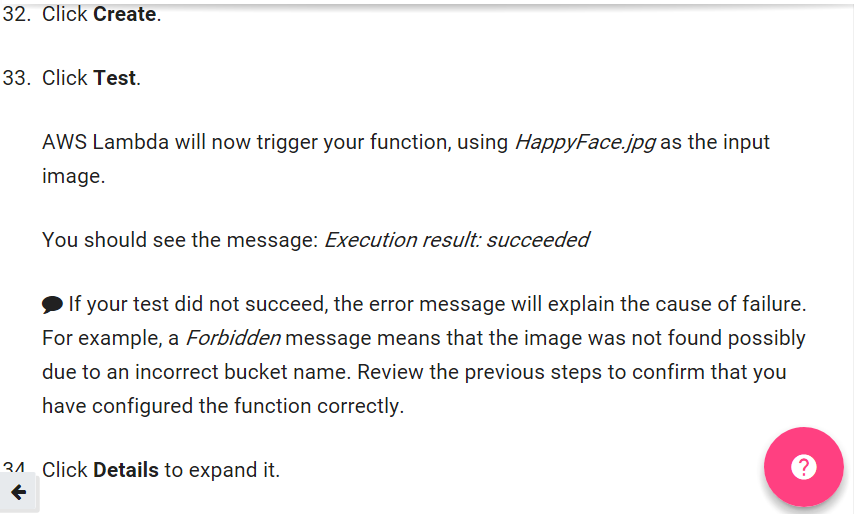


S3link: <https://s3.amazonaws.com/us-west-2-aws-training/awsu-spl/spl-88/scripts/CreateThumbnail.zip>

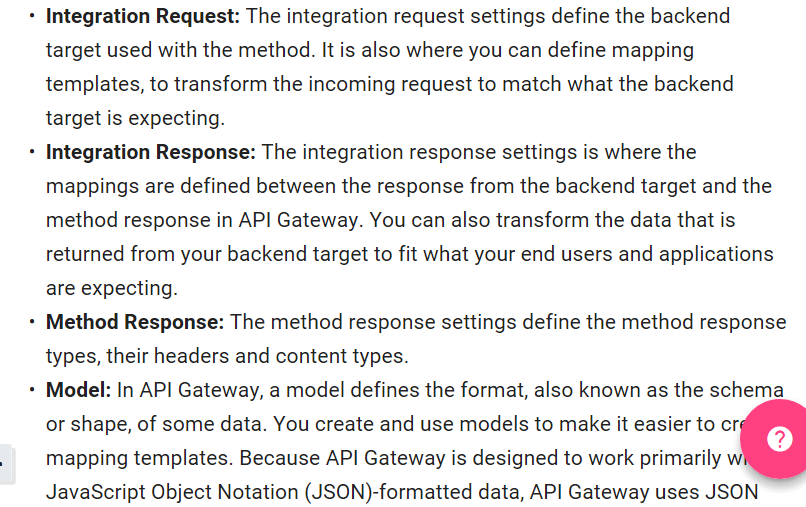
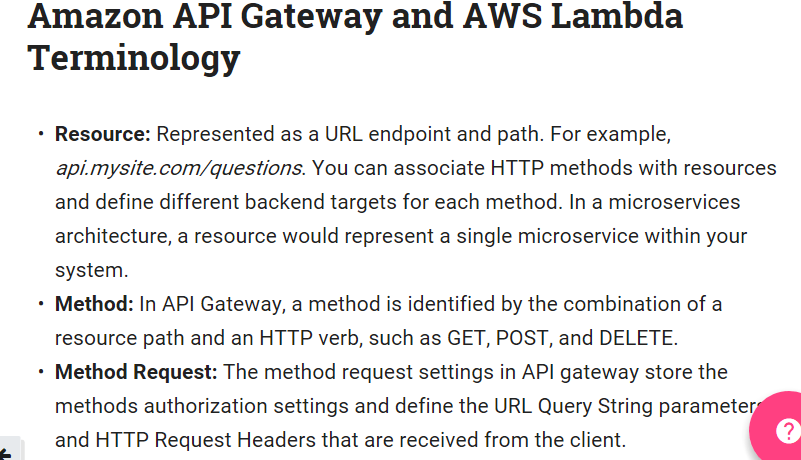
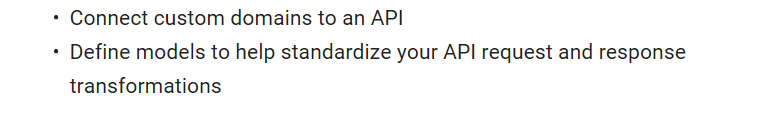
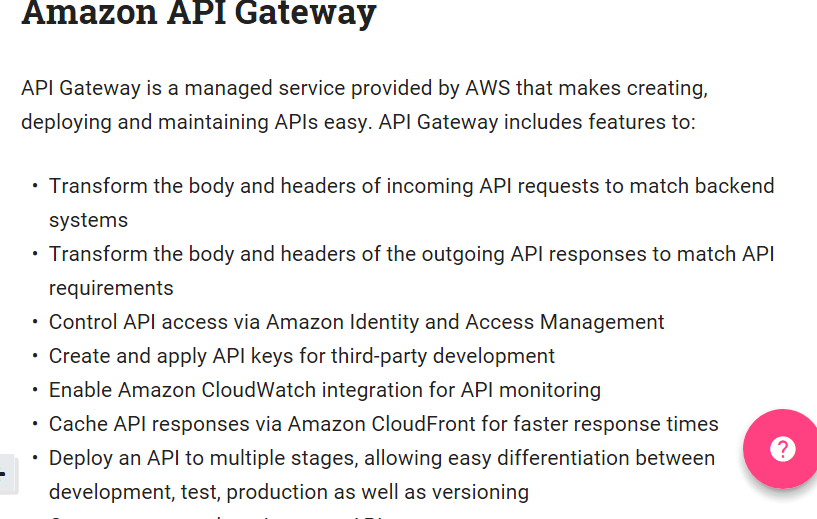
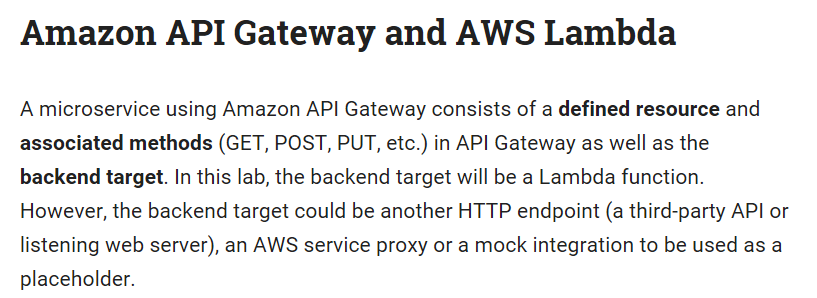
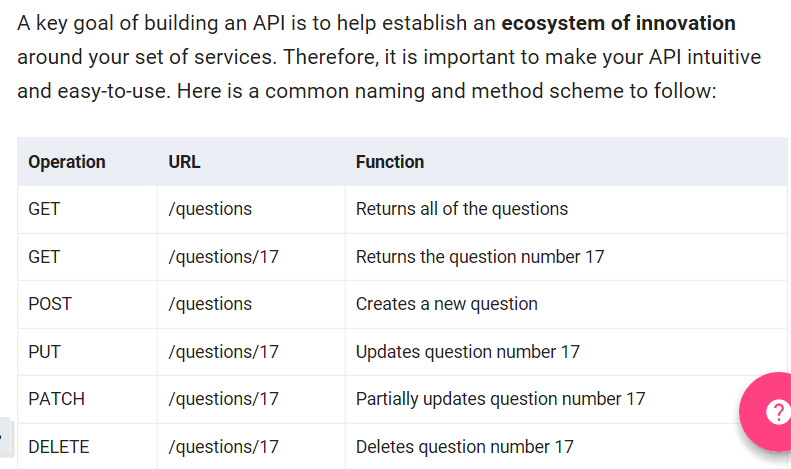
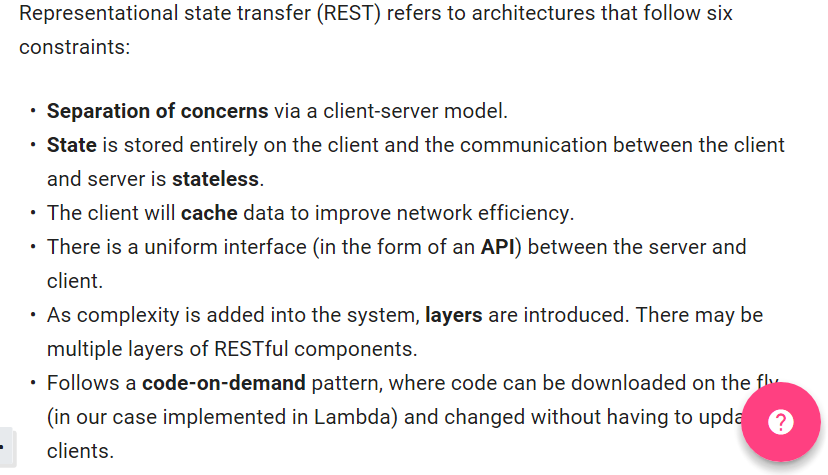
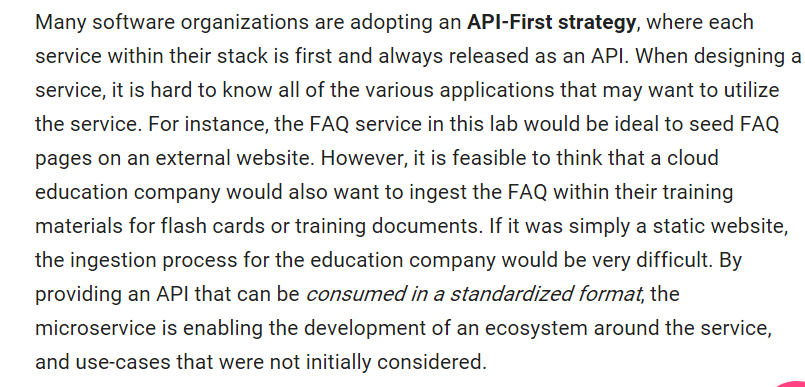
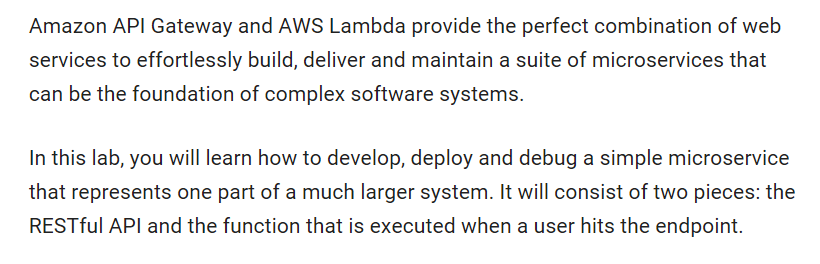
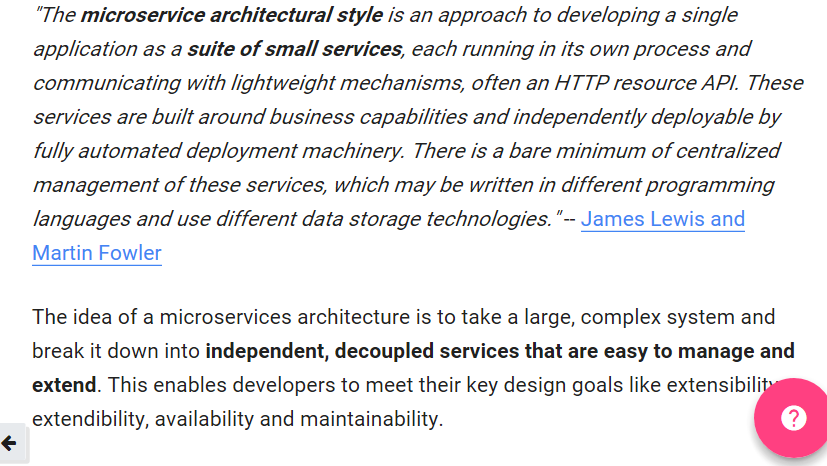
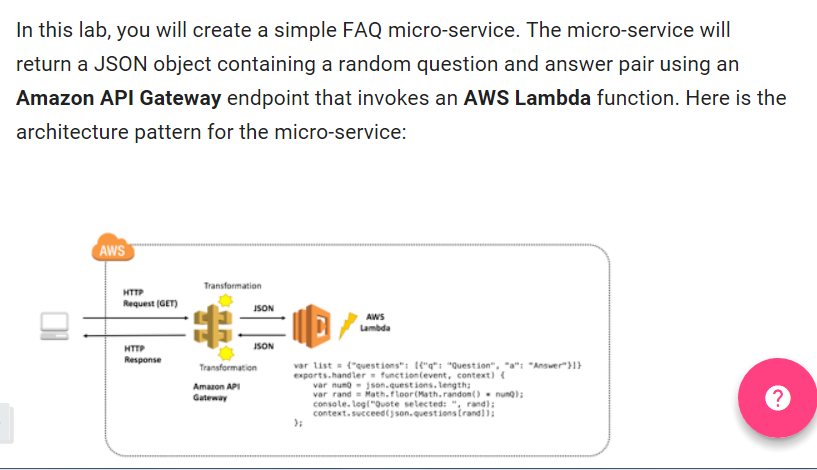




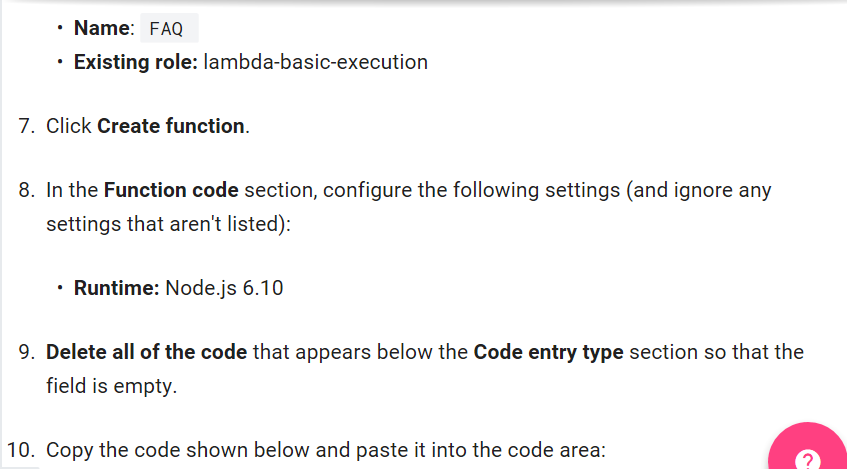
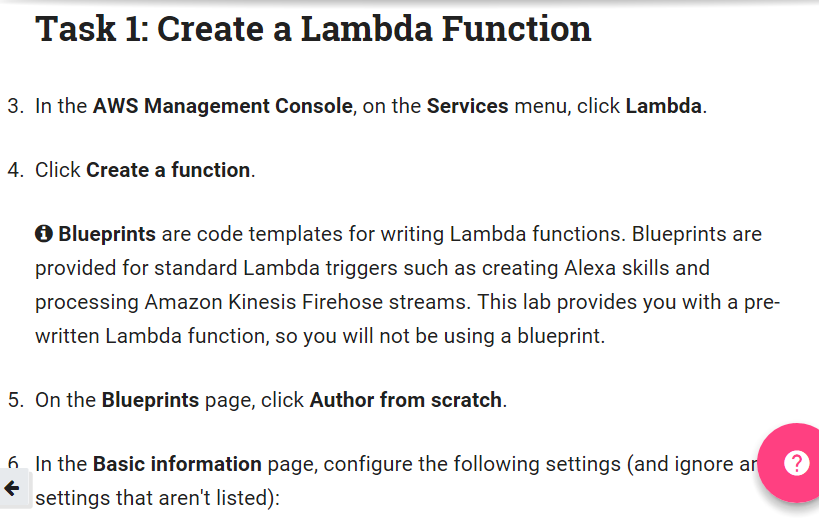




**Amazon API Gateway**

****

**Example of standard rest service:** [**https://github.com/WhiteHouse/api-standards#pragmatic-rest**](https://github.com/WhiteHouse/api-standards#pragmatic-rest)



var json = {

"service": "lambda",

"reference": "https://aws.amazon.com/lambda/faqs/",

"questions": [{

"q": "What is AWS Lambda?",

"a": "AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app."

},{

"q":"What events can trigger an AWS Lambda function?",

"a":"You can use AWS Lambda to respond to table updates in Amazon DynamoDB, modifications to objects in Amazon S3 buckets, logs arriving in Amazon CloudWatch logs, incoming emails to Amazon Simple Email Service, notifications sent from Amazon SNS, messages arriving in an Amazon Kinesis stream, client data synchronization events in Amazon Cognito, and custom events from mobile applications, web applications, or other web services. You can also invoke a Lambda function on a defined schedule using the AWS Lambda console."

},{

"q":"When should I use AWS Lambda versus Amazon EC2?",

"a":"Amazon Web Services offers a set of compute services to meet a range of needs. Amazon EC2 offers flexibility, with a wide range of instance types and the option to customize the operating system, network and security settings, and the entire software stack, allowing you to easily move existing applications to the cloud. With Amazon EC2 you are responsible for provisioning capacity, monitoring fleet health and performance, and designing for fault tolerance and scalability. AWS Elastic Beanstalk offers an easy-to-use service for deploying and scaling web applications in which you retain ownership and full control over the underlying EC2 instances. Amazon EC2 Container Service is a scalable management service that supports Docker containers and allows you to easily run distributed applications on a managed cluster of Amazon EC2 instances. AWS Lambda makes it easy to execute code in response to events, such as changes to Amazon S3 buckets, updates to an Amazon DynamoDB table, or custom events generated by your applications or devices. With Lambda you do not have to provision your own instances; Lambda performs all the operational and administrative activities on your behalf, including capacity provisioning, monitoring fleet health, applying security patches to the underlying compute resources, deploying your code, running a web service front end, and monitoring and logging your code. AWS Lambda provides easy scaling and high availability to your code without additional effort on your part."

},{

"q":"What kind of code can run on AWS Lambda?",

"a":"AWS Lambda offers an easy way to accomplish many activities in the cloud. For example, you can use AWS Lambda to build mobile back-ends that retrieve and transform data from Amazon DynamoDB, handlers that compress or transform objects as they are uploaded to Amazon S3, auditing and reporting of API calls made to any Amazon Web Service, and server-less processing of streaming data using Amazon Kinesis."

},{

"q":"What languages does AWS Lambda support?",

"a":"AWS Lambda supports code written in Node.js (JavaScript), Python, and Java (Java 8 compatible). Your code can include existing libraries, even native ones. Lambda functions can easily launch processes using languages supported by Amazon Linux, including Bash, Go, and Ruby. Please read our documentation on using Node.js, Python and Java."

},{

"q":"Can I access the infrastructure that AWS Lambda runs on?",

"a":"No. AWS Lambda operates the compute infrastructure on your behalf, allowing it to perform health checks, apply security patches, and do other routine maintenance."

},{

"q":"How does AWS Lambda isolate my code?",

"a":"Each AWS Lambda function runs in its own isolated environment, with its own resources and file system view. AWS Lambda uses the same techniques as Amazon EC2 to provide security and separation at the infrastructure and execution levels."

},{

"q":"How does AWS Lambda secure my code?",

"a":"AWS Lambda stores code in Amazon S3 and encrypts it at rest. AWS Lambda performs additional integrity checks while your code is in use."

},{

"q":"What is an AWS Lambda function?",

"a":"The code you run on AWS Lambda is uploaded as a Lambda function. Each function has associated configuration information, such as its name, description, entry point, and resource requirements. The code must be written in a stateless style i.e. it should assume there is no affinity to the underlying compute infrastructure. Local file system access, child processes, and similar artifacts may not extend beyond the lifetime of the request, and any persistent state should be stored in Amazon S3, Amazon DynamoDB, or another Internet-available storage service. Lambda functions can include libraries, even native ones."

},{

"q":"Will AWS Lambda reuse function instances?",

"a":"To improve performance, AWS Lambda may choose to retain an instance of your function and reuse it to serve a subsequent request, rather than creating a new copy. Your code should not assume that this will always happen."

},{

"q":"What if I need scratch space on disk for my AWS Lambda function?",

"a":"Each Lambda function receives 500MB of non-persistent disk space in its own /tmp directory."

},{

"q":"Why must AWS Lambda functions be stateless?",

"a":"Keeping functions stateless enables AWS Lambda to rapidly launch as many copies of the function as needed to scale to the rate of incoming events. While AWS Lambda's programming model is stateless, your code can access stateful data by calling other web services, such as Amazon S3 or Amazon DynamoDB."

},{

"q":"Can I use threads and processes in my AWS Lambda function code?",

"a":"Yes. AWS Lambda allows you to use normal language and operating system features, such as creating additional threads and processes. Resources allocated to the Lambda function, including memory, execution time, disk, and network use, must be shared among all the threads/processes it uses. You can launch processes using any language supported by Amazon Linux."

},{

"q":"What restrictions apply to AWS Lambda function code?",

"a":"Lambda attempts to impose few restrictions on normal language and operating system activities, but there are a few activities that are disabled: Inbound network connections are managed by AWS Lambda, only TCP/IP sockets are supported, and ptrace (debugging) system calls are restricted. TCP port 25 traffic is also restricted as an anti-spam measure."

},{

"q":"How do I create an AWS Lambda function using the Lambda console?",

"a":"You can author the code for your function using the inline editor in the AWS Lambda console. You can also package the code (and any dependent libraries) as a ZIP and upload it using the AWS Lambda console from your local environment or specify an Amazon S3 location where the ZIP file is located. Uploads must be no larger than 50MB (compressed). You can use the AWS Eclipse plugin to author and deploy Lambda functions in Java and Node.js. If you are using Node.js, you can author the code for your function using the inline editor in the AWS Lambda console. Go to the console to get started."

},{

"q":"How do I create an AWS Lambda function using the Lambda CLI?",

"a":"You can package the code (and any dependent libraries) as a ZIP and upload it using the AWS CLI from your local environment, or specify an Amazon S3 location where the ZIP file is located. Uploads must be no larger than 50MB (compressed). Visit the Lambda Getting Started guide to get started."

},{

"q":"Which versions of Python are supported?",

"a":"Lambda provides a Python 2.7-compatible runtime to execute your Lambda functions. Lambda will include the latest AWS SDK for Python (boto3) by default."

},{

"q":"How do I compile my AWS Lambda function Java code?",

"a":"You can use standard tools like Maven or Gradle to compile your Lambda function. Your build process should mimic the same build process you would use to compile any Java code that depends on the AWS SDK. Run your Java compiler tool on your source files and include the AWS SDK 1.9 or later with transitive dependencies on your classpath. For more details, see our documentation."

},{

"q":"What is the JVM environment Lambda uses for execution of my function?",

"a":"Lambda provides the Amazon Linux build of openjdk 1.8."

}

]

}

exports.handler = function(event, context) {

var rand = Math.floor(Math.random() \* json.questions.length);

console.log("Quote selected: ", rand);

var response = {

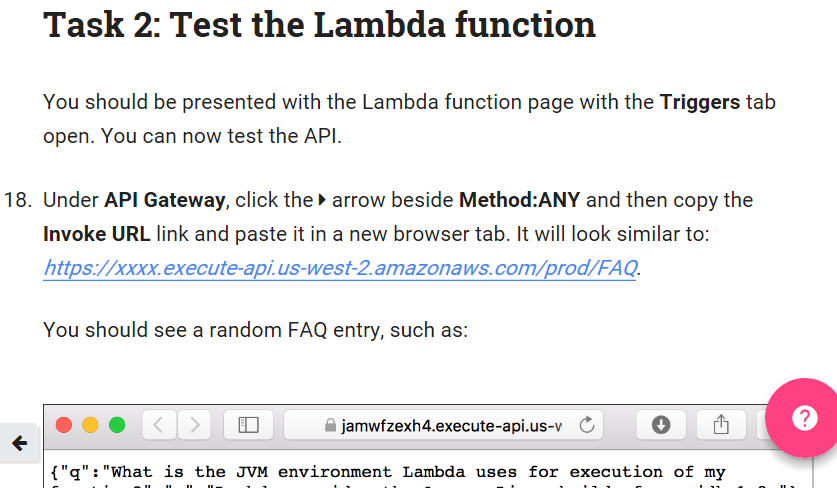
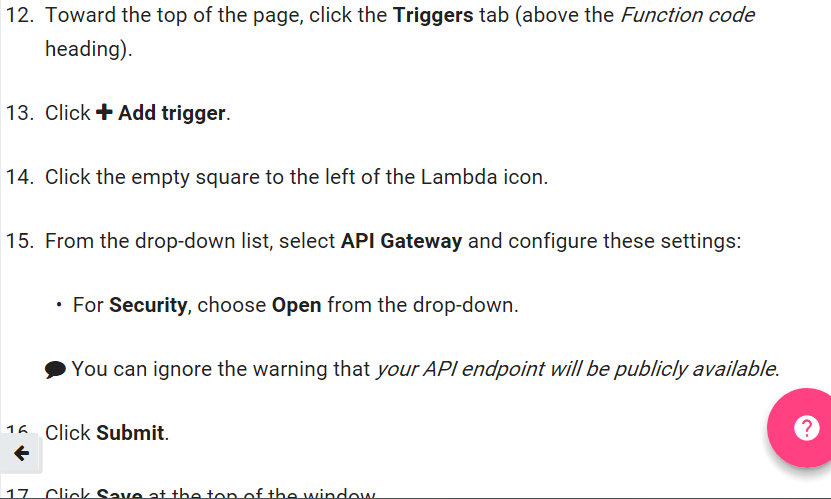
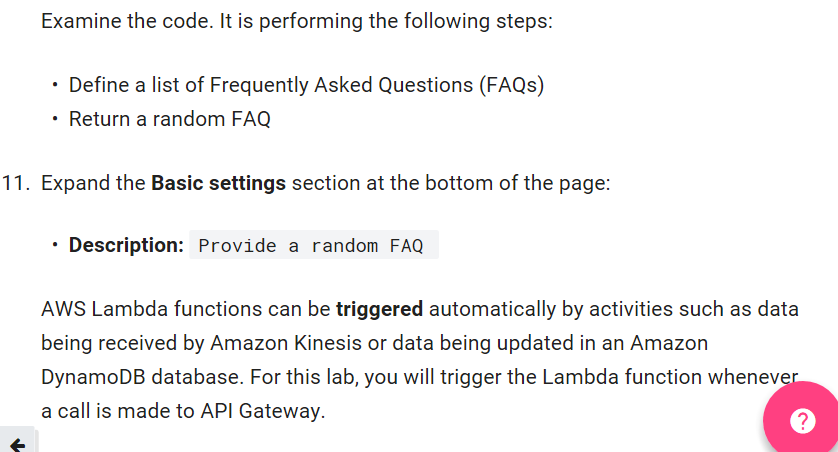
body: JSON.stringify(json.questions[rand])

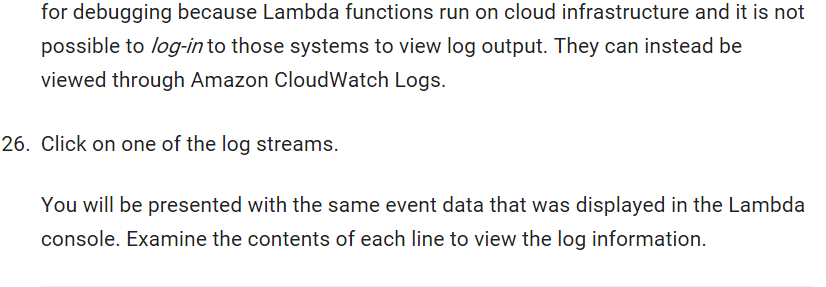
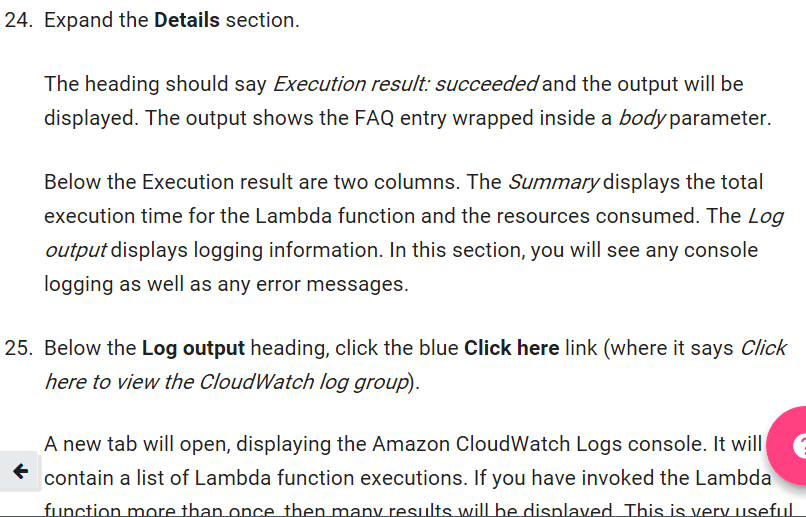
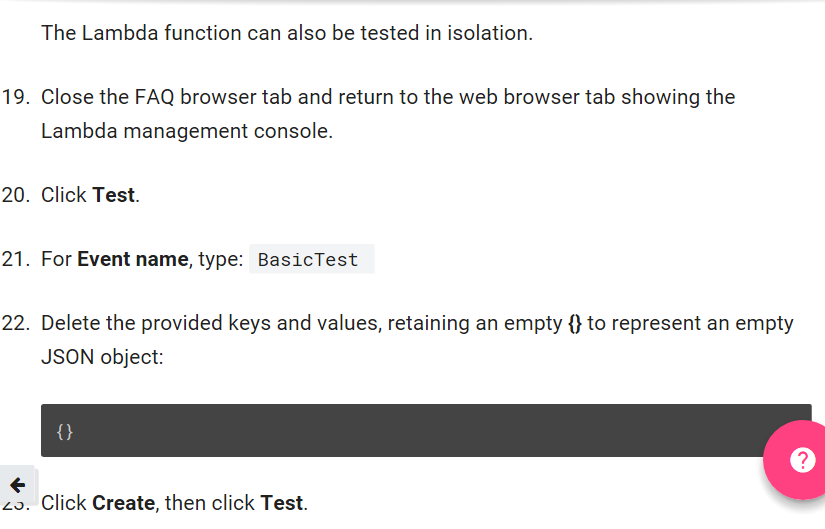
};

console.log(response);

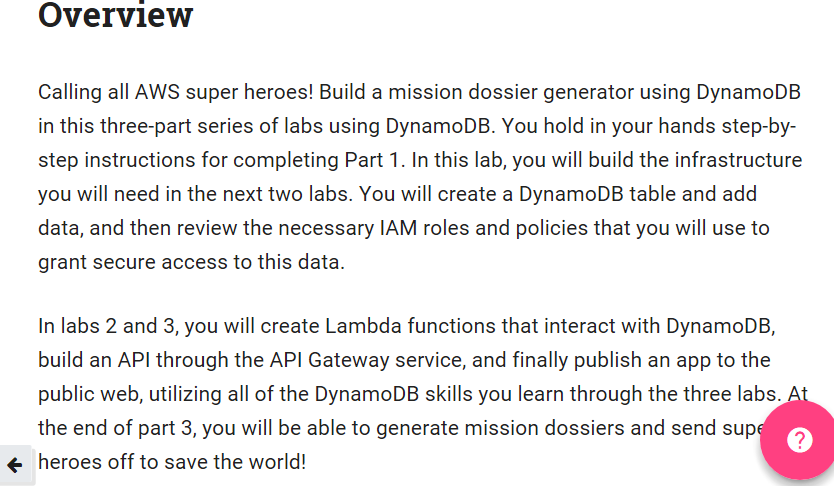
context.succeed(response);

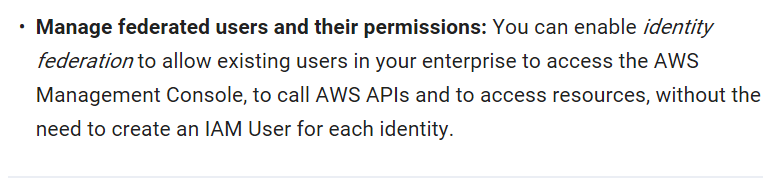
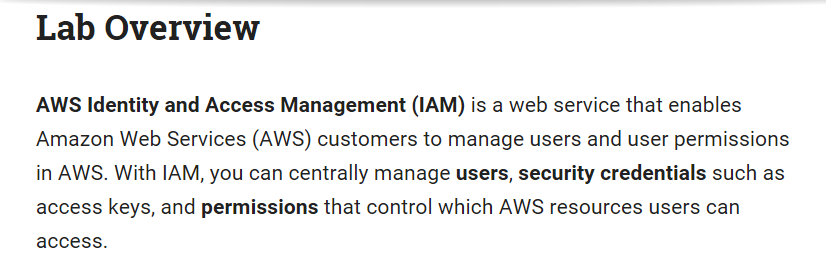
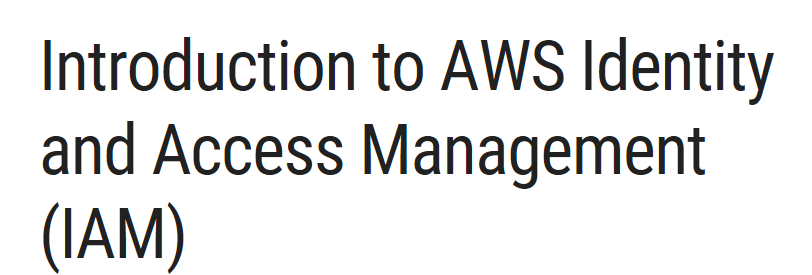
};





**Amazon api documentation:** [**http://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html**](http://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html)

****

****

**Introduction to AWS Identity and Access Management (IAM)**

Lab Overview

AWS Identity and Access Management (IAM) is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.

•Exploring pre-created IAM Users and Groups

•Inspecting IAM policies as applied to the pre-created Groups

•Following a real-world scenario, adding Users to Groups with specific capabilities enabled

•Locating and using the IAM sign-in URL

•Experimenting with the effects of policies on service access

Other AWS Services

During this lab, you may receive error messages when performing actions beyond the steps in this lab guide. These messages will not impact your ability to complete the lab.

What is AWS Identity and Access Management?

AWS Identity and Access Management (IAM) can be used to:

•Manage IAM Users and their access: You can create Users and assign them individual security credentials (access keys, passwords, and multi-factor authentication devices). You can manage permissions to control which operations a User can perform.

•Manage IAM Roles and their permissions: An IAM Role is similar to a User, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a Role is intended to be assumable by anyone who needs it.

•Manage federated users and their permissions: You can enable identity federation to allow existing users in your enterprise to access the AWS Management Console, to call AWS APIs and to access resources, without the need to create an IAM User for each identity.

Start Lab

Task 1: Explore the Users and Groups

In this task, you will explore the Users and Groups that have already been created for you in IAM.

3.In the AWS Management Console, click Services then click IAM (Identity & Access Management).

4.In the left navigation pane, click Users.

Some IAM Users have already been created for you:

•user-1

•user-2

•user-3

There is also an awsstudent user, which you can ignore for this lab.

5.Click on the name user-1 to reveal details about the User.

6.In the User details, find the following facts about this User:

•It currently has no policies attached (Permissions tab)

•It is not a member of any group (Groups tab)

•It has a password assigned already (Security credentials tab)

7.In the left navigation pane, click Groups.

The following groups have already been created for you:

•EC2-Admin

•EC2-Support

•S3-Support

8.Click on the group name for EC2-Support.

9.Click the Permissions tab.

This group has a Managed Policy associated with it, called AmazonEC2ReadOnlyAccess. Managed Policies are pre-built policies (built either by AWS or by your administrators) that can be attached to IAM Users and Groups. When the policy is updated, the changes immediately apply against all Users and Groups that are attached to the policy.

10.Under Actions, click Show Policy.

A policy defines what actions are allowed or denied for specific AWS resources. This policy is granting permission to List and Describe information about EC2, Elastic Load Balancing, CloudWatch and Auto Scaling. This ability to view resources, but not modify them, is ideal for assigning to a Support role.

Note the basic structure of the statements in the IAM Policies:

• Effect says whether to Allow or Deny the permissions.

• Action specifies the API calls that can be made against an AWS Service (eg cloudwatch:ListMetrics).

• Resource defines the scope of entities covered by the policy rule (eg a specific Amazon S3 bucket or Amazon EC2 instance, or \* which means any resource).

11.Close the dialog box, click Groups again and look at the permissions assigned to the S3-Support group.

This policy has permissions to Get and List resources in Amazon S3.

12.Close the dialog box, click Groups again and look at the permissions assigned to the EC2-Admin group.

This Group is different. Instead of a Managed Policy, it has an Inline Policy, which is a policy assigned to just one User or Group. Inline Policies are used to override standard permissions for specific situations.

13.Under Actions, click Edit Policy to view the policy.

The policy is granting permission to view (Describe) information about Amazon EC2 and also the ability to Start and Stop instances.

14.Click Cancel to close the policy.

Business Scenario

For the remainder of this lab, you will work with these Users and Groups to enable permissions supporting the following business scenario:

Your company is growing its use of Amazon Web Services, and is using many Amazon EC2 instances and a great deal of Amazon S3 storage. You wish to give access to new staff depending upon their job function:

User

In Group

Permissions

user-1 S3-Support Read-Only access to Amazon S3

user-2 EC2-Support Read-Only access to Amazon EC2

user-3 EC2-Admin View, Start and Stop Amazon EC2 instances

Task 2: Add Users to Groups

You have recently hired user-1 into a role where they will provide support for Amazon S3. You will add them to the S3-Support group so that they inherit the necessary permissions via the attached AmazonS3ReadOnlyAccess policy.

 You can ignore any "not authorized" errors that appear during this task. They are caused by your lab account having limited permissions and will not impact your ability to complete the lab.

15.In the left navigation pane, click Groups.

16.Click on the S3-Support group name.

17.In the Users tab, click Add Users to Group.

18.Select user-1 and click the blue Add Users button in the lower-right.

You have hired user-2 into a role where they will provide suport for Amazon EC2.

19.Using similar steps to above, add user-2 to the EC2-Support group.

You have also hired user-3 as your Amazon EC2 administrator, who manage your EC2 instances.

20.Using simlar steps to above, add user-3 to the EC2-Admin group.

21.When you are finished, again click Groups in the left navigation pane. Each Group should have a 1 in the Users column for the number of Users in each Group.

If you do not have a 1 beside each group, revisit the above instructions to ensure that each user is assigned to a Group, as shown in the table in the Business Scenario section.

Task 3: Sign-In and Test Users

In this task, you will test the permissions of each IAM User.

22.In the left navigation pane, click Dashboard.

An IAM users sign-in link is displayed It will look similar to: https://123456789012.signin.aws.amazon.com/console

This link can be used to sign-in to the AWS Account you are currently using.

23.Copy the IAM users sign-in link.

24.Paste a copy of the sign-in link in a text editor because you will be using it again several times.

25.Paste the link into your web browser address bar and hit Enter. This will take you to the Sign-In screen.

 If you receive a message stating "You must first log out before logging into a different AWS account", then click the "To logout, click here" link, then paste the link into your web browser address bar again and hit Enter. This will take you to the Sign-In screen.

You will now sign-in as user-1, who has been hired as your Amazon S3 storage support staff.

26.Sign-in with:

•IAM user name:

•Password:

27.From the main AWS Console, click Services and then S3.

28.Click the name of a bucket and browse the contents.

Your user is part of the S3-Support Group in IAM, so they have permission to view a list of Amazon S3 buckets and their contents.

Now, test whether they have access to Amazon EC2.

29.From the main AWS Console, click Services and then EC2.

30.In the left navigation pane, click Instances.

You cannot see any instances! Instead, it says You are not authorized to perform this operation. This is because your user has not been assigned any permissions to use Amazon EC2.

You will now sign-in as user-2, who has been hired as your Amazon EC2 support person.

31.Paste the sign-in link into your web browser address bar again. If it is not in your clipboard, retrieve it from the text editor where you stored it earlier.

32.Sign-in with:

•IAM user name:

•Password:

33.From the main AWS Console, click Services and then EC2.

34.In the left navigation pane, click Instances.

You are now able to see an Amazon EC2 instance because you have Read Only permissions. However, you will not be able to make any changes to Amazon EC2 resources.

 If you cannot see an Amazon EC2 instance, then your Region may be incorrect. In the top-right of the screen, pull-down the Region menu and select the region that you noted at the start of the lab (eg Oregon).

35.Select the EC2 instance.

36.In the Actions menu, under Instance State, click Stop.

37.Click Yes, Stop.

You will receive an error stating You are not authorized to perform this operation. This demonstrates that the policy is only permitted you to view information, without making changes.

Next, check whether they can access Amazon S3.

38.From the main AWS Console, click Services and then S3.

You receive an Access Denied error because this User does not permission to use Amazon S3.

You will now sign-in as user-3, who has been hired as your Amazon EC2 administrator.

39.Paste the sign-in link into your web browser address bar again. If it is not in your clipboard, retrieve it from the text editor where you stored it earlier.

40.Sign-in with:

•IAM user name:

•Password:

41.From the main AWS Console, click Services and then EC2.

42.In the left navigation pane, click Instances.

As an EC2 Administrator, you should now have permissions to Stop the Amazon EC2 instance.

43.Select the EC2 instance.

 If you cannot see an Amazon EC2 instance, then your Region may be incorrect. In the top-right of the screen, pull-down the Region menu and select the region that you noted at the start of the lab (eg Oregon).

44.In the Actions menu, under Instance State, click Stop.

45.Click Yes, Stop.

The instance will enter the stopping state and will shutdown.