



Deploy a Web Application in AWS Elastic Kubernetes Service

Kawtar Oukil



Table des matières

I. Race Objectives	2
Task 1: Understand the entire project elements and scope as well as be able to describe basic aspects of Amazon Elastic Kubernetes Services	2
Task 2: Set all permission needed for our project, create roles and policies.....	3
Task 3: Create an Elastic Container Registry to store our container images, and set some identities configuration need for the project	21
Task 4: Clone an application from GitHub, build a Docker image, run a container and push the image to Elastic Container Registry	25
Task 5: Create an Elastic Kubernetes Cluster and set all configuration needed	30
Task 6: Create and set a Node Group to host containers in a POD architecture	34
Task 7: Deploy an application in Elastic Kubernetes Cluster using an Image hosted in Elastic Container Registry.....	38
Task 8: Scale the Node Group to support the increase of new pods in the Kubernetes cluster.....	49
Task 9: Clean Up the environment.....	60
II. QCM.....	62



I. Race Objectives

In this project, we are going to focus on **three** learning objectives:

1. *Build a container image and deploy it to Elastic Container Registry*
2. *Create an Elastic Kubernetes cluster and deploy a web application*
3. *Scale the Elastic Kubernetes Cluster*

One of the most popular methods to deploy a container is using Kubernetes. Kubernetes is a continued orchestrator with a lot of features to deploy and manage thousands of containers in a fault tolerance in high scalability environment, like an open-source project, Kubernetes is managed by Cloud Native Foundation and this is used by a lot of companies and is being sold by many vendors as a managed service.

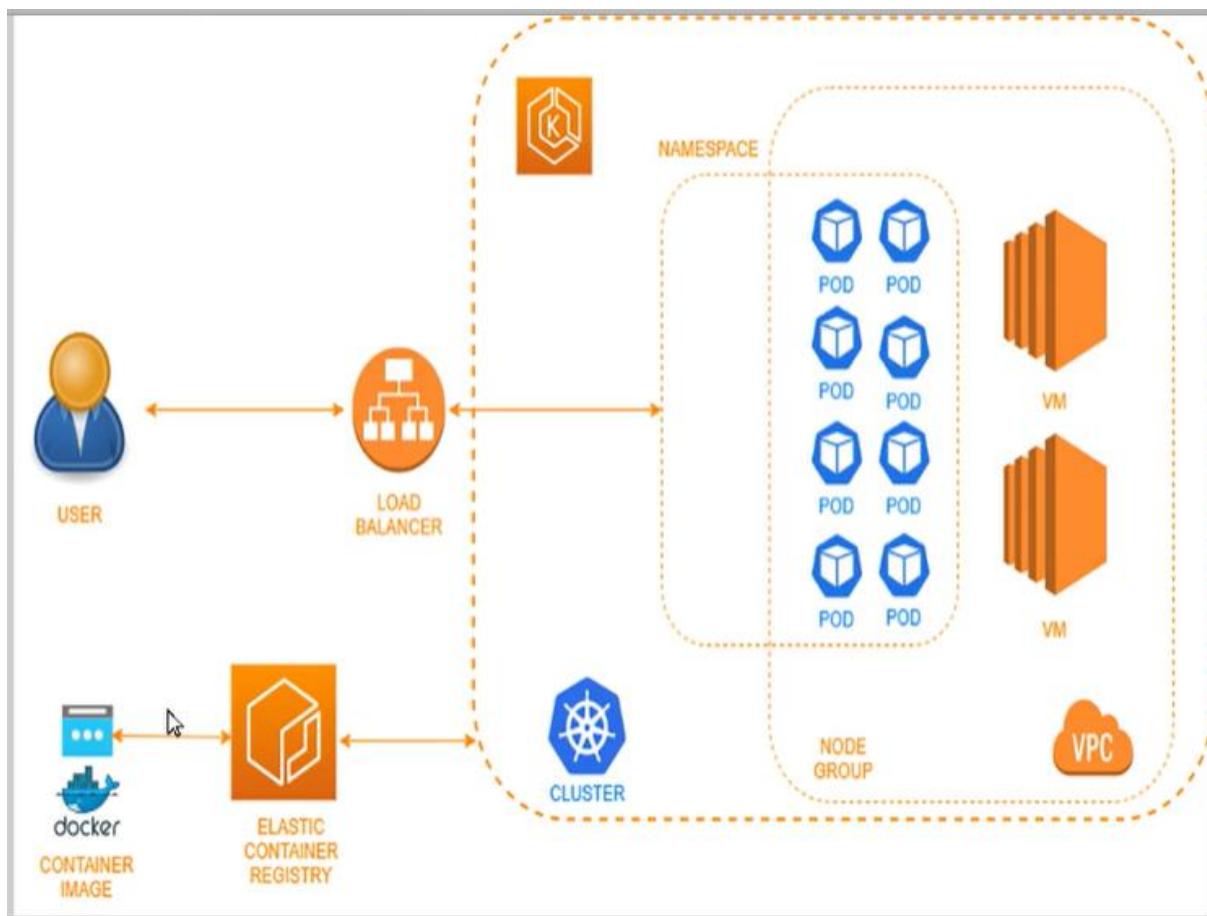
Amazon Web Services has a set of management services that provides an entire ecosystem to create, store and deploy containers in Kubernetes.

The main services are elastic Kubernetes services, which are Kubernetes offered as a service.

In this project we will understand how to clone a web application project, build a Docker container image uploaded to AWS and deployed into Elastic Kubernetes servicing.

Task 1: Understand the entire project elements and scope as well as be able to describe basic aspects of Amazon Elastic Kubernetes Services

In this first task, we are going to see our project architecture:



In this diagram above, we see the elements that we will create in AWS. Some elements will be created for us and some by AWS based on the choosing configuration on the bottom, we can see we have a container image for our application. We will build the image and push it to elastic container has history which will be our depository for our image. Later we will use this image started in Alaska container history for deployment in our Kubernetes cluster. We will also create a Kubernetes cluster with one node group. This node group, we have at least two our machines and we will be able to have thousands of pods in this empty group.

How elastic Kubernetes services is a managed services will create for us other elements in the VPC networking where the virtual machines will be deployed like the load balancer to support the end user in grass and other configurations.

Task 2: Set all permission needed for our project, create roles and policies

The first thing we need is to create one AWS access key:



AWS Management Console

AWS services

- All services

Build a solution

Get started with simple wizards and automated workflows.

Launch a virtual machine With EC2. 2-3 minutes

Build a web app With Elastic Beanstalk. 6 minutes

Explore AWS

Modernize Your APIs with GraphQL

AWS AppSync is a fully-managed GraphQL service that improves app performance and developer productivity.

Identity and Access Management (IAM)

- Dashboard
- Access management
 - User groups
 - Users
 - Roles
 - Policies
 - Identity providers
 - Account settings
- Access reports
 - Access analyzer
 - Archive rules
 - Analyzers
 - Settings
- Credential report

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS Global Support.

- >Password
- Multi-factor authentication (MFA)
- Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend ! If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make sure you store it in a secure place.

Created	Access Key ID	Last Used	Last Used Region	Last Used Service

[Create New Access Key](#)

Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access to AWS services, consider creating a new IAM user with limited permissions and generating access keys for that user instead. [Learn more](#)

And we open Access Area, and we create a new key using the creating the 'creating new access key' button



Screenshot of the AWS IAM 'Your Security Credentials' page. A modal window titled 'Create Access Key' is open, showing a success message: 'Your access key (access key ID and secret access key) has been created successfully.' It also instructs to download the key file. Below the modal is a table with columns: Created, Access Key ID, Last Used, Last Used Region, and Last Used Service. A button 'Create New Access Key' is visible at the bottom.

Then the access key was created successfully. And later, we click on 'Download Key File' button to save the credentials to our cloud desktop. This key give access to our AWS services, that's why it is recommended to delete this key after finishing our project.

Now, we need to create some roles and policies to be used in Kubernetes services

In the left of the menu, we select 'Policies':

Screenshot of the AWS IAM 'Policies' page. The 'Create policy' button is highlighted. A table is shown with columns: Policy name, Type, and Used as. A status message 'Loading...' is displayed below the table.

And the we click on 'Create Policy' to create a new policy:



The screenshot shows the AWS IAM 'Create policy' interface. At the top, there's a navigation bar with the AWS logo, a search bar, and links for 'Services', 'Coursera Universidade Global', 'Global', and 'Support'. Below the navigation is a title 'Create policy' and a progress bar with three steps: 1 (blue), 2 (white), and 3 (white). A vertical 'Documentation' sidebar is on the left. The main area is titled 'Visual editor' with a 'JSON' tab next to it. It includes buttons for 'Expand all' and 'Collapse all'. A large central panel is labeled 'Select a service' with a 'Clone | Remove' button at the top right. Below this are sections for 'Service', 'Actions', 'Resources', and 'Request conditions', each with a 'Choose a service...' link. The JSON tab contains the following code:

```
1 "Statement": [
2   {
3     "Action": [
4       "autoscaling:DescribeAutoScalingGroups",
5       "autoscaling:DescribeAutoScalingInstances",
6       "autoscaling:DescribeLaunchConfigurations",
7       "autoscaling:DescribeTags",
8       "autoscaling:SetDesiredCapacity",
9       "autoscaling:TerminateInstanceInAutoScalingGroup",
10      "ec2:DescribeLaunchTemplateVersions"
11    ],
12    "Resource": "*",
13    "Effect": "Allow"
14  }
15]
```

And here, in the JSON space, we gonna add a code:

The screenshot shows the same 'Create policy' interface in JSON mode. The JSON tab is selected, and the code area contains the following JSON policy:

```
1 "Statement": [
2   {
3     "Action": [
4       "autoscaling:DescribeAutoScalingGroups",
5       "autoscaling:DescribeAutoScalingInstances",
6       "autoscaling:DescribeLaunchConfigurations",
7       "autoscaling:DescribeTags",
8       "autoscaling:SetDesiredCapacity",
9       "autoscaling:TerminateInstanceInAutoScalingGroup",
10      "ec2:DescribeLaunchTemplateVersions"
11    ],
12    "Resource": "*",
13    "Effect": "Allow"
14  }
15]
```

At the bottom of the JSON area, there are status indicators: 'Security: 0', 'Errors: 0', 'Warnings: 0', and 'Suggestions: 0'.

Next we click on: 'next Text':



Policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor **JSON** Import managed policy

```
3 Statement": [
4   {
5     "Action": [
6       "autoscaling:DescribeAutoScalingGroups",
7       "autoscaling:DescribeAutoScalingInstances",
8       "autoscaling:DescribeLaunchConfigurations",
9       "autoscaling:DescribeTags",
10      "autoscaling:SetDesiredCapacity",
11      "autoscaling:TerminateInstanceInAutoScalingGroup",
12      "ec2:DescribeLaunchTemplateVersions"
13    ],
14    "Resource": "*",
15    "Effect": "Allow"
16  }
17]
18}
```

Character count: 356 of 6,144. [Cancel](#) [Next: Tags](#)

And 'Next Review':

Create policy

1 **2** 3

Add tags (Optional)
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add tag](#)
You can add up to 50 more tags

[Cancel](#) [Previous](#) [Next: Preview](#)

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

After, we add a name to the policy, and we click on 'create policy':



aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Maximum 1000 characters. Use alphanumeric and '+-=,@-_ ' characters.

Summary Filter

Service	Access level	Resource
Allow (2 of 284 services) Show remaining 282		
EC2	Limited: List	All resources
EC2 Auto Scaling	Full: Read Limited: List, Write	All resources

Tags Key Value

No tags associated with the resource.

* Required Cancel Previous Create policy

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Similarly to policy, we gonna create a role, then we gone on ‘roles’ option in the menu, and we click on ‘create role’ button:

aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Identity and Access Management (IAM)

Create role Delete role

Showing 9 results

Role name	Trusted entities	Last activity
AWServi...	AWS service: eks (Service-Linked role)	Today
AWServi...	AWS service: eks-nodegroup (Service-Link...	4 days
AWServi...	AWS service: autoscaling (Service-Linked role)	4 days
AWServi...	AWS service: ecs (Service-Linked role)	155 days
AWServi...	AWS service: elasticloadbalancing (Service-...	4 days
AWServi...	AWS service: organizations (Service-Linked r...	None
AWServi...	AWS service: support (Service-Linked role)	None
AWServi...	AWS service: trustedadvisor (Service-Linked ...	None
ecsTaskE...	AWS service: ecs-tasks	155 days

Then we choose the AWS service option:



aws Services ▾ Search for services, features, marketp. [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Create role

1 2 3

Select type of trusted entity

AWS service EC2, Lambda and others Another AWS account Belonging to you or 3rd party Web identity Cognito or any OpenID provider SAML 2.0 federation Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose a use case

Common use cases

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

API Gateway CodeBuild EMR Containers IoT SiteWise RDS

* Required Cancel Next: Permissions

And, in the use case we select the EKS option:

aws Services ▾ Search for services, features, marketp. [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Application Auto Scaling	Data Lifecycle Manager	Forecast	MQ	SageMaker
Application Discovery Service	Data Pipeline	GameLift	Machine Learning	Security Hub
Batch	DataBrew	Global Accelerator	Macie	Service Catalog
Braket	DataSync	Glue	Managed Blockchain	Step Functions
Budgets	DeepLens	Greengrass	MediaConvert	Storage Gateway
Certificate Manager	Directory Service	GuardDuty	Migration Hub	Systems Manager
Chime	DynamoDB	Health Organizational View	Network Firewall	Textract
CloudFormation	EC2	Honeycode	OpsWorks	Transfer
CloudHSM	EC2 - Fleet	IAM Access Analyzer	Personalize	Trusted Advisor
CloudTrail	EC2 Auto Scaling	Incident Manager	Purchase Orders	VPC
CloudWatch Alarms	EC2 Image Builder	Inspector	QLDB	WorkLink
CloudWatch Application Insights	EKS	IoT	RAM	WorkMail
CloudWatch Events	EMR			

Later, we select 'EKS cluster':



aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

CloudWatch Alarms EKS IoT RAM WorkMail

CloudWatch Application Insights EMR

CloudWatch Events

Select your use case

EKS
Allows EKS to manage clusters on your behalf.

EKS - Cluster
Allows access to other AWS service resources that are required to operate clusters managed by EKS.

EKS - Fargate pod
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.

EKS - Fargate profile
Allows EKS to run Fargate tasks.

EKS - Nodegroup
Allow EKS to manage nodegroups on your behalf.

* Required

Cancel

Next: Permissions

And, then click ‘Next Permission’, and then we add a name to the role, and we click on ‘create role’:

aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Create Role Review

Provide the required information below and review this role before you create it.

Role name* courseraeeks_role

Use alphanumeric and '+=.,@-_ ' characters. Maximum 64 characters.

Role description Allows access to other AWS service resources that are required to operate clusters managed by EKS.

Maximum 1000 characters. Use alphanumeric and '+=.,@-_ ' characters.

Trusted entities AWS service: eks.amazonaws.com

Policies AmazonEKSClusterPolicy

Permissions boundary Permissions boundary is not set

No tags were added.

* Required Cancel Previous Create role

And, then our role was created successfully:



aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Identity and Access Management (IAM)

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analyzers

Settings

Credential report

Organization activity

The role courseraeeks_role has been created.

Create role Delete role

Search Showing 10 results

Role name	Trusted entities	Last activity
AWServi...	AWS service: eks (Service-Linked role)	Today
AWServi...	AWS service: eks-nodegroup (Service-Linked role)	4 days
AWServi...	AWS service: autoscaling (Service-Linked role)	4 days
AWServi...	AWS service: ecs (Service-Linked role)	155 days
AWServi...	AWS service: elasticloadbalancing (Service-Linked role)	4 days
AWServi...	AWS service: organizations (Service-Linked role)	None
AWServi...	AWS service: support (Service-Linked role)	None
AWServi...	AWS service: trustedadvisor (Service-Linked role)	None
courserae...	AWS service: eks	None

In this project, we will to create a second role, with another option of use case called 'EC2',

aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Service	Service	Service	Service	Service
Application Auto Scaling	Data Lifecycle Manager	Forecast	MQ	SageMaker
Application Discovery Service	Data Pipeline	GameLift	Machine Learning	Security Hub
Batch	DataBrew	Global Accelerator	Macie	Service Catalog
Braket	DataSync	Glue	Managed Blockchain	Step Functions
Budgets	DeepLens	Greengrass	MediaConvert	Storage Gateway
Certificate Manager	Directory Service	GuardDuty	Migration Hub	Systems Manager
Chime	DynamoDB	Health Organizational View	Network Firewall	Textract
CloudFormation	EC2	Honeycode	OpsWorks	Transfer
CloudHSM	EC2 - Fleet	IAM Access Analyzer	Personalize	Trusted Advisor
CloudTrail	EC2 Auto Scaling	Incident Manager	Purchase Orders	VPC
CloudWatch Alarms	EC2 Image Builder	Inspector	QLDB	WorkLink
CloudWatch Application Insights	EKS	IoT	RAM	WorkMail
CloudWatch Events	EMR			

* Required Cancel Next: Permissions



aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾ Select your use case

EC2
Allows EC2 instances to call AWS services on your behalf.

EC2 - Scheduled Instances
Allows EC2 Scheduled Instances to manage instances on your behalf.

EC2 - Spot Fleet
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

EC2 - Spot Fleet Auto Scaling
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

EC2 - Spot Fleet Tagging
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.

EC2 - Spot Instances
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.

EC2 Role for AWS Systems Manager
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

EC2 Spot Fleet Role
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

* Required Cancel Next: Permissions

Later, we need to choose some policies, the first one is **AmazonEKS worker node policy**:

aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Filter policies ▾ Showing 1 result
amazonEKSworerno

Policy name	Used as
<input checked="" type="checkbox"/> AmazonEKSWorkerNodePolicy	None

- ▶ Set permissions boundary

* Required Cancel Previous Next: Tags

Next one is **AmazonEC2ContainerRegistryReadOnly**:



aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Filter policies ▾ Showing 1 result

	Policy name ▾	Used as
	AmazonEC2ContainerRegistryReadOnly	None

Set permissions boundary

* Required Cancel Previous Next: Tags

And the next one is **AmazonEKS_CNI_Policy**:

aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Filter policies ▾ Showing 1 result

	Policy name ▾	Used as
	AmazonEKS_CNI_Policy	None

Set permissions boundary

* Required Cancel Previous Next: Tags

And now, the next one is our policy created a few seconds ago, called '**coursera_policy**':



aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Filter policies ▾ coursera_policy Showing 1 result

	Policy name ▾	Used as
<input checked="" type="checkbox"/>	coursera_policy	None

Set permissions boundary

* Required Cancel Previous Next: Tags

And then we click on 'next tags' -> 'next review', and confirm if our 4 policies are in the list, and we will give a name of the role, and click on 'create role':

aws Services ▾ Search for services, features, marketpl [Alt+S] Coursera Universidade Global ▾ Global ▾ Support ▾

Create role Review

Provide the required information below and review this role before you create it.

Role name* courseranode_role

Use alphanumeric and '+=.,@-_ ' characters. Maximum 64 characters.

Role description Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=.,@-_ ' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies

- AmazonEKSWorkerNodePolicy
- AmazonEC2ContainerRegistryReadOnly
- AmazonEKS_CNI_Policy
- coursera_policy

* Required Cancel Previous Create role

Then our role was created successfully:



The screenshot shows the AWS Identity and Access Management (IAM) service. In the left sidebar, under 'Identity and Access Management (IAM)', the 'Roles' option is selected. A success message at the top right says 'The role courseranode_role has been created.' Below it, there are two buttons: 'Create role' and 'Delete role'. A search bar labeled 'Search' is present. The main area displays a table titled 'Showing 11 results' with columns for 'Role name', 'Trusted entities', and 'Last activity'. The table lists various AWS service roles, including 'AWS service: eks', 'AWS service: eks-nodegroup', 'AWS service: autoscaling', 'AWS service: ecs', 'AWS service: elasticloadbalancing', 'AWS service: organizations', 'AWS service: support', 'AWS service: trustedadvisor', and 'courserae...', all of which have 'eks' as their trusted entity.

Then, now we need to create an EC2 key pair, it is used to authenticating EC2 to off our machines and it is used to authenticate in our node group later if we needed. So, to do that we need to go to the institute dashboard page on the search bar, and we click on '**search on institute**' to open the institute dashboard:

The screenshot shows the AWS search interface. The search bar at the top contains the query 'ec2'. The results are displayed under the heading 'Services'. The first result is 'EC2' with the subtext 'Virtual Servers in the Cloud'. Below it are three other services: 'EC2 Image Builder', 'AWS Compute Optimizer', and 'AWS Firewall Manager'. On the left sidebar, there are links for 'Services (6)', 'Features (34)', 'Documentation (225,545)', and 'Marketplace (1,233)'. At the bottom of the screen, there is a footer with the text 'Footer content'.

EC2 is a region basic in services in AWS:



The screenshot shows the AWS EC2 console interface. On the left, there's a navigation sidebar with options like 'EC2 Dashboard', 'Instances' (expanded to show 'Instances', 'Instance Types', 'Launch Templates', etc.), and 'Images'. The main area has a blue header bar with the text 'Welcome to the new EC2 console!'. Below it, a message says: 'We're redesigning the EC2 console to make it easier to use and improve performance. We'll release new screens periodically. We encourage you to try them and let us know where we can make improvements. To switch between the old console and the new console, use the New EC2 Experience toggle.' A 'Resources' section displays various Amazon EC2 resources in the US East (N. Virginia) Region, all with a value of 0: Instances (running), Dedicated Hosts, Elastic IPs, Instances, Key pairs, Load balancers, Placement groups, Security groups, Snapshots, and Volumes. At the bottom, a tooltip suggests using the AWS Launch Wizard for Microsoft SQL Server Always On availability groups.

The first thing we need to do is to choose the region where we will create our resources in this project. On top right we gonna choose our region

This screenshot is similar to the previous one but focuses on the region selection. The 'N. Virginia' region is selected, highlighted in yellow. A dropdown menu lists other regions: 'US East (N. Virginia)' (selected), 'US East (Ohio)', 'US West (N. California)', 'US West (Oregon)', 'Africa (Cape Town)', 'Asia Pacific (Hong Kong)', 'Asia Pacific (Mumbai)', 'Asia Pacific (Osaka)', 'Asia Pacific (Seoul)', 'Asia Pacific (Singapore)', 'Asia Pacific (Sydney)', 'Asia Pacific (Tokyo)', 'Canada (Central)', and 'Europe (Frankfurt)'. The rest of the interface is identical to the first screenshot, showing the 'Resources' section with zero values for all categories.

We will choose the US East.

Later, in institute dashboard, in the left menu, we gone to key pairs option below network and security, and we click on '**key pairs**':



Scheduled Instances
Capacity Reservations
▼ Images
AMIs
▼ Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces New
▼ Load Balancing
Load Balancers
Target Groups New

Welcome to the new EC2 console! We're redesigning the EC2 console to make it easier to use and improve performance. We'll release new screens periodically. We encourage you to try them and let us know where we can make improvements. To switch between the old console and the new console, use the New EC2 Experience toggle.

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0	Dedicated Hosts	0
Elastic IPs	0	Instances	0
Key pairs	0	Load balancers	0
Placement groups	0	Security groups	1
Snapshots	0	Volumes	0

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. [Learn more](#)

Now, we click on 'create key pair':

Scheduled Instances
Capacity Reservations
▼ Images
AMIs
▼ Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces New
▼ Load Balancing
Load Balancers
Target Groups New

Key pairs

Create key pair

Filter key pairs

Name	Fingerprint	ID
No key pairs to display		

Now, we choose a name to our key pair, and we select the format, and then we click on 'create key pair':



aws Services ▾ Search for services, features, marketp [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support ▾

Create Key pair

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

File format

pem
For use with OpenSSH

ppk
For use with PuTTY

Tags (Optional)

No tags associated with the resource.

Add tag

You can add 50 more tags.

Create key pair

So, now we have our key pair created:

aws Services ▾ Search for services, features, marketp [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support ▾

New EC2 Experience Tell us what you think X Successfully created key pair X

EC2 Dashboard New

Events

Tags

Limits

Instances Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances New

Dedicated Hosts

Scheduled Instances

Capacity Reservations

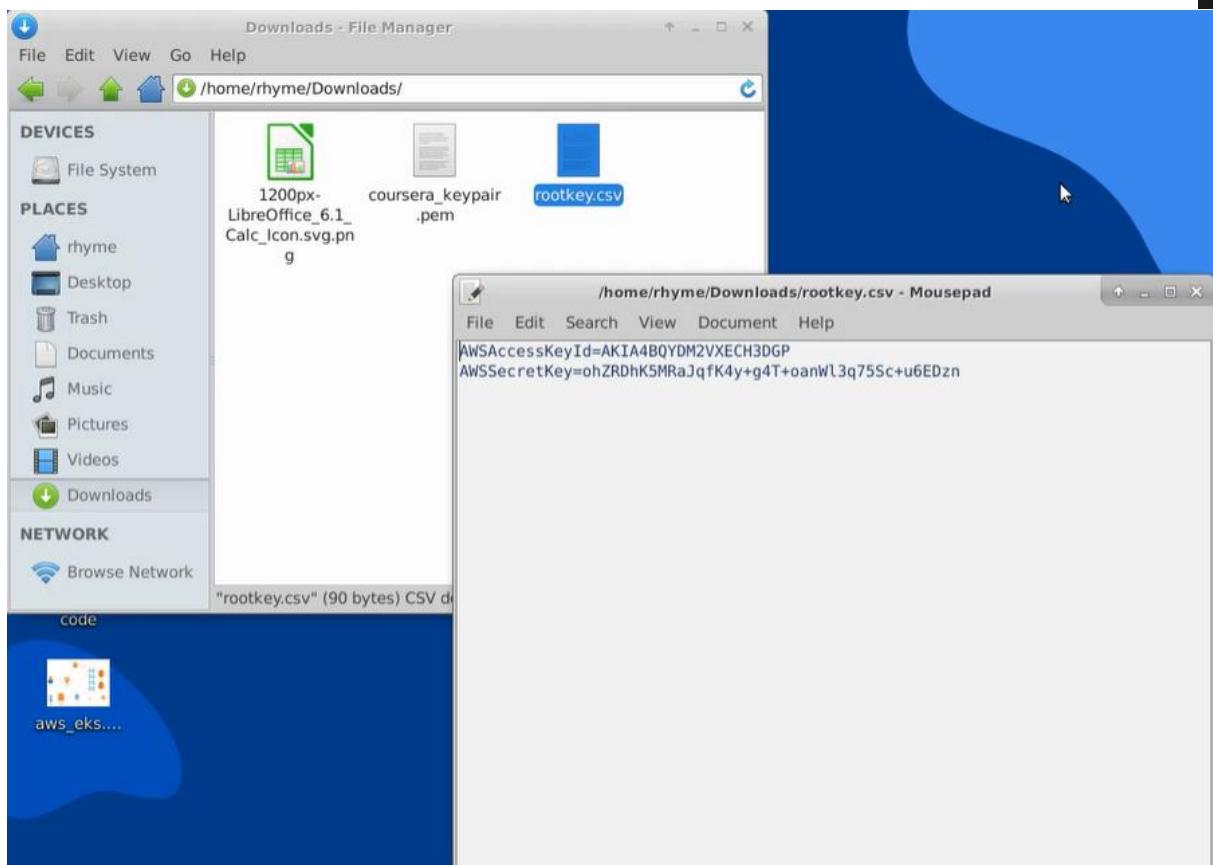
Key pairs (1)

Name	Fingerprint	ID
coursera_keypair	c5:aa:c0:c7:c8:4e:7c:a5:bd:a4:6c:77:63:...	key-0908490cc...

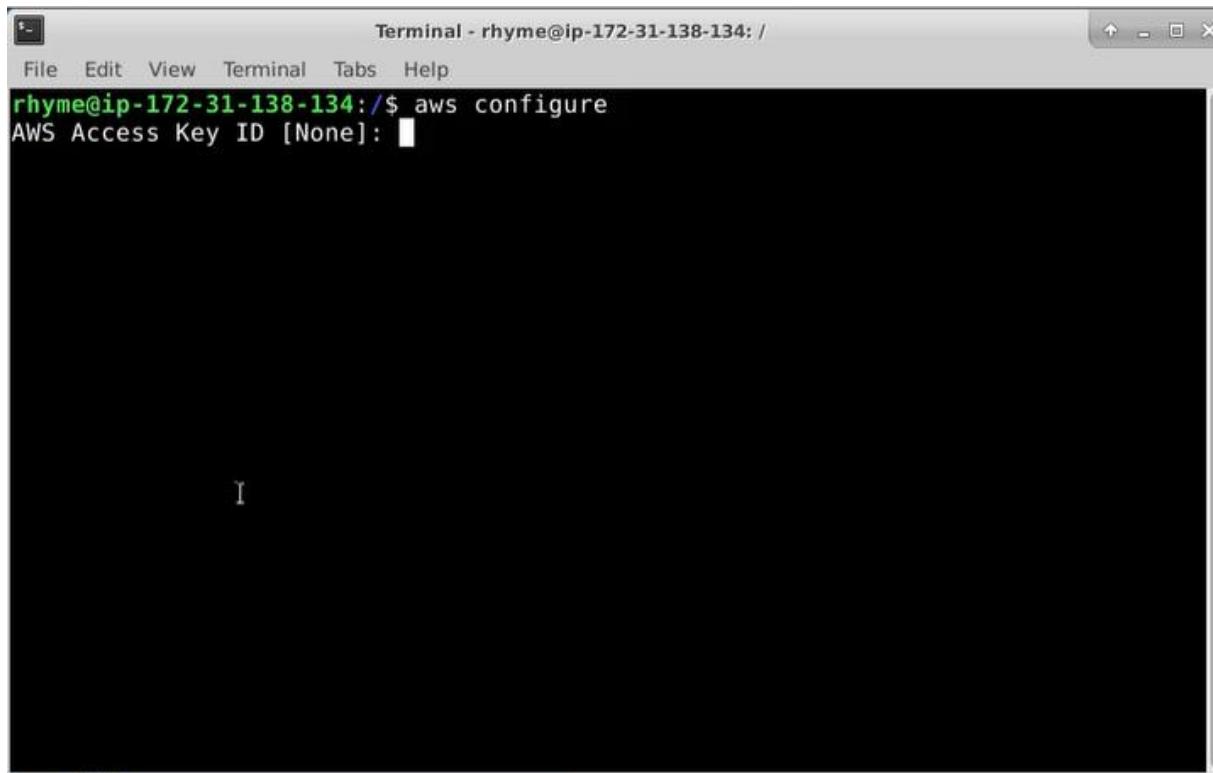
Create key pair

Let's now configure the AWS credentials in command line interface.

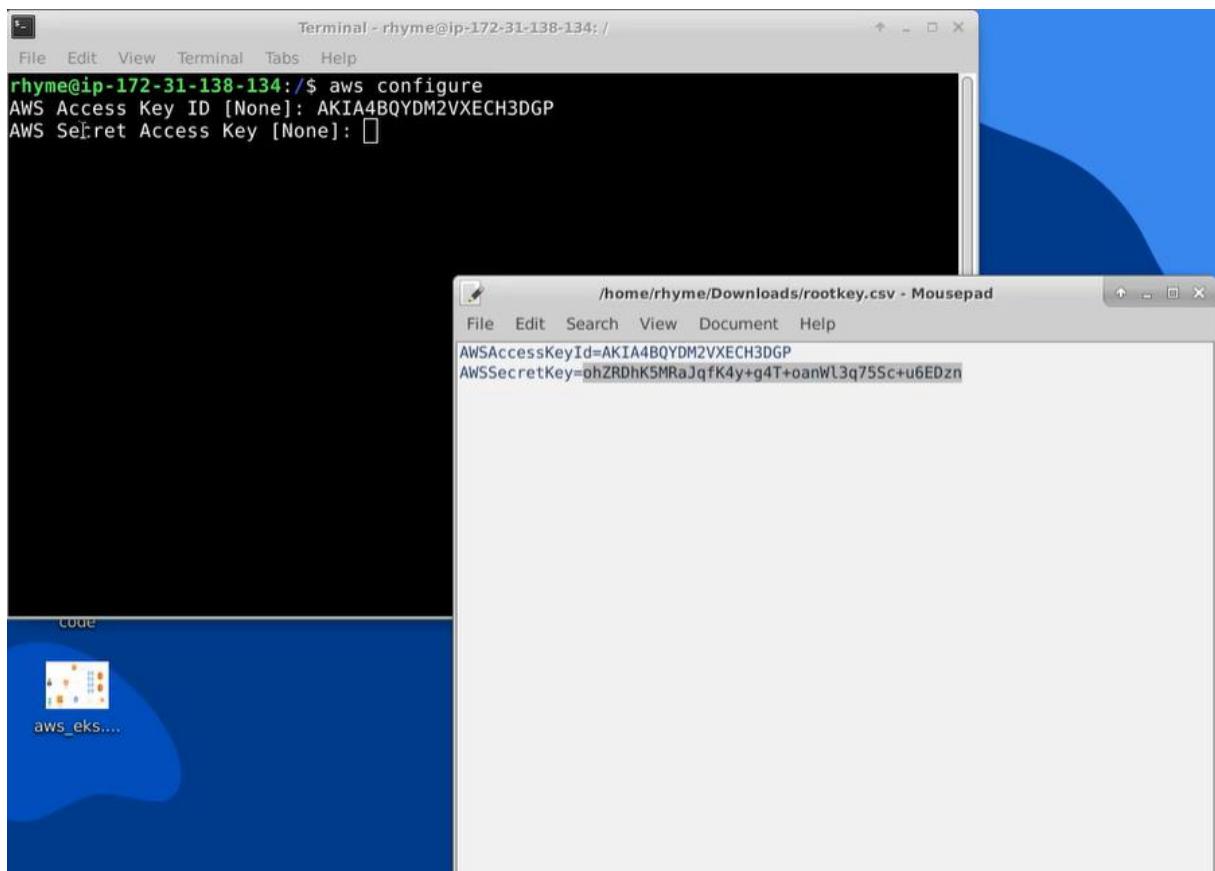
AWS common interface is already installed and we our access keys created and downloaded; it is called '**rootkey.csv**':



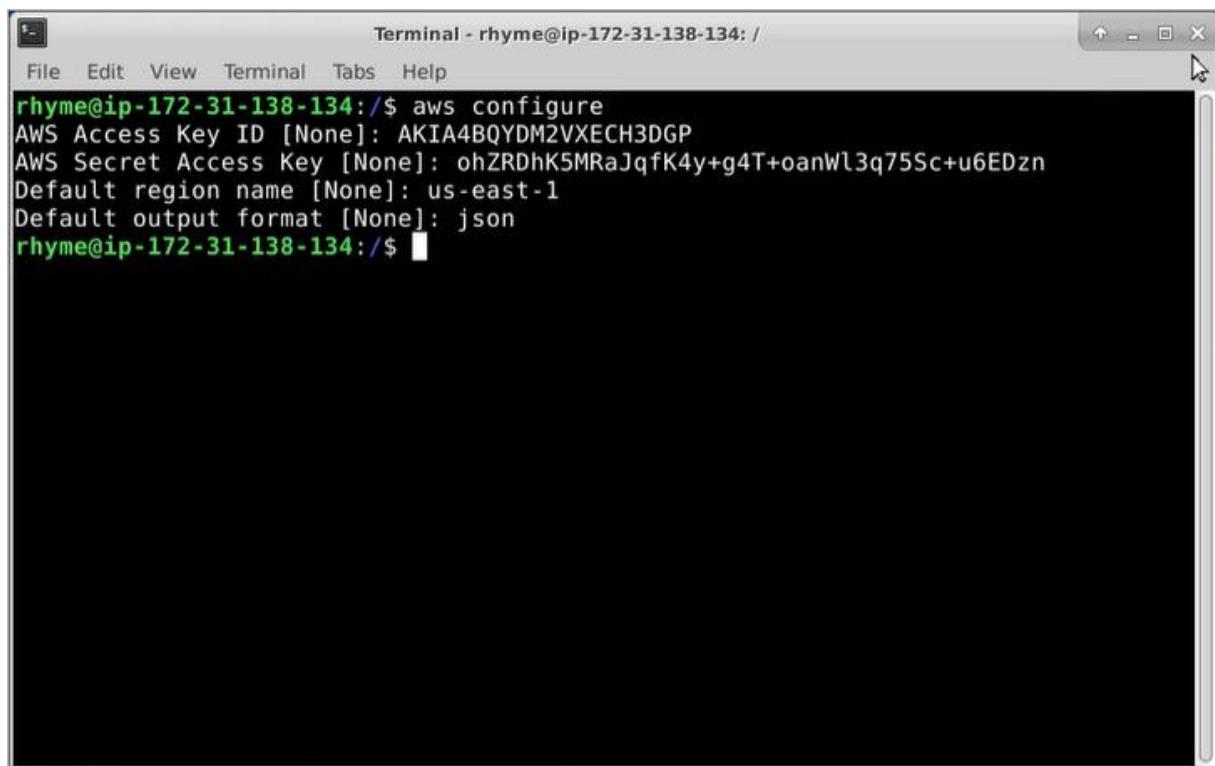
We gone to the terminal and we type the command: #aws configure:



And here we paste the key ID and the Secret Access key from the csv file:



Later, we type the region name and the output format:



Finally, all thing is configured.



Task 3: Create an Elastic Container Registry to store our container images, and set some identities configuration need for the project

Elastic Container Registry (ECR) is the place where we can create our major repositories I public or private mode. In this project we will used to store our web application container image in AWS console.

On the search bar on the top of the page of AWS console:

The screenshot shows the AWS Management Console homepage. At the top, there's a navigation bar with the AWS logo, a search bar, and links for Services, Coursera Universidad Global, N. Virginia, and Support. Below the navigation bar, the title "AWS Management Console" is displayed. On the left, there's a sidebar titled "AWS services" with sections for "Recently visited services" (EC2, IAM) and "All services". The main content area features several cards: "Build a solution" with options like "Launch a virtual machine" (With EC2, 2-3 minutes), "Build a web app" (With Elastic Beanstalk, 6 minutes), "Build using virtual servers" (With Lightsail, 1-2 minutes), and "Register a domain" (With Route 53, 3 minutes). To the right, there's a box for "Stay connected to your AWS resources on-the-go" about the AWS Console Mobile App, and another box for "Explore AWS" with links to "Free AWS Training", "Modernize Your APIs with GraphQL", and "AWS Certification". At the bottom, there are links for Feedback, English (US), and various AWS terms like Privacy Policy, Terms of Use, and Cookie preferences.

we gonna search for ECR and click on **Alaska Container history**:



Screenshot of the AWS search results page for 'ecr'. The search bar at the top contains 'ecr'. The left sidebar shows 'Services (6)', 'Features (4)', 'Documentation (30,978)', and 'Marketplace (12)'. The main content area displays the following services:

- Secrets Manager**: Easily rotate, manage, and retrieve secrets throughout their lifecycle.
- Elastic Container Service**: Highly secure, reliable, and scalable way to run containers.
- Elastic Container Registry**: Fully-managed Docker container registry : Share and deploy container software, publ...

The 'Top features' section includes:

- Secrets**: Secrets Manager feature
- Parameter Store**: Systems Manager feature

The 'Registries' section includes:

- Key Management Service**: Securely Generate and Manage AWS Encryption Keys

A sidebar on the right provides information about AWS resources and certification.

Then, we got this page:

Screenshot of the Amazon Elastic Container Registry (ECR) landing page. The title is 'Amazon Elastic Container Registry' and the subtitle is 'Share and deploy container software, publicly or privately'. A subtext states: 'Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share, and deploy your container images and artifacts anywhere.' A 'Create a repository' button is visible. The 'How it works' section shows a flow from writing code to running containers. The 'Benefits' section lists 'Fully managed' and 'Secure'. The 'Pricing (US)' section notes that pay only for stored data and internet transfers. The 'Getting started' section links to 'What is Amazon ECR?', 'Getting started with ECR', and 'Set up a CI/CD pipeline with ECR'.

Then now in this page, we go to the left menu, and below Amazon ECR, we click on **Repositories**:



Screenshot of the AWS Amazon Elastic Container Registry (ECR) landing page.

The left sidebar shows navigation for Amazon Container Services, with "Amazon ECR" and "Repositories" selected. The main content area features the title "Amazon Elastic Container Registry" and the subtitle "Share and deploy container software, publicly or privately". It includes a brief description of ECR as a fully managed container registry and a diagram titled "How it works" showing the flow from code to container images to deployment. A "Create a repository" button is prominently displayed.

We check that we are in the AWS region we are choosing for this project:

Screenshot of the AWS ECR Repositories page.

The left sidebar shows navigation for Amazon Container Services, with "Amazon ECR" and "Repositories" selected. The main content area shows a list of "Private repositories" with no results found. To the right, a sidebar lists various AWS regions with their corresponding endpoint URLs, such as "US East (N. Virginia) us-east-1", "US East (Ohio) us-east-2", and "Middle East (Bahrain) me-south-1".

On clicking on '**Create Repository**', we got this page below, And there we add the name of our repository, and make the visibility settings to **private**:



Screenshot of the AWS ECR 'Create repository' page.

General settings

Visibility settings Info
Choose the visibility setting for the repository.
 Private
Access is managed by IAM and repository policy permissions.
 Public
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
827905500843.dkr.ecr.us-east-1.amazonaws.com/ **website**

7 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

Tag immutability Info
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
 Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be triggered manually.

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

After, we click on ‘Create Repository’ button, and then the repository was created successfully:

Screenshot of the AWS ECR 'Repositories' page showing the newly created repository.

Amazon Container Services

Amazon ECS
Clusters
Task definitions

Amazon EKS
Clusters

Amazon ECR
Repositories
Registries
Public gallery

ECR > Repositories

Private **Public**

Private repositories (1)

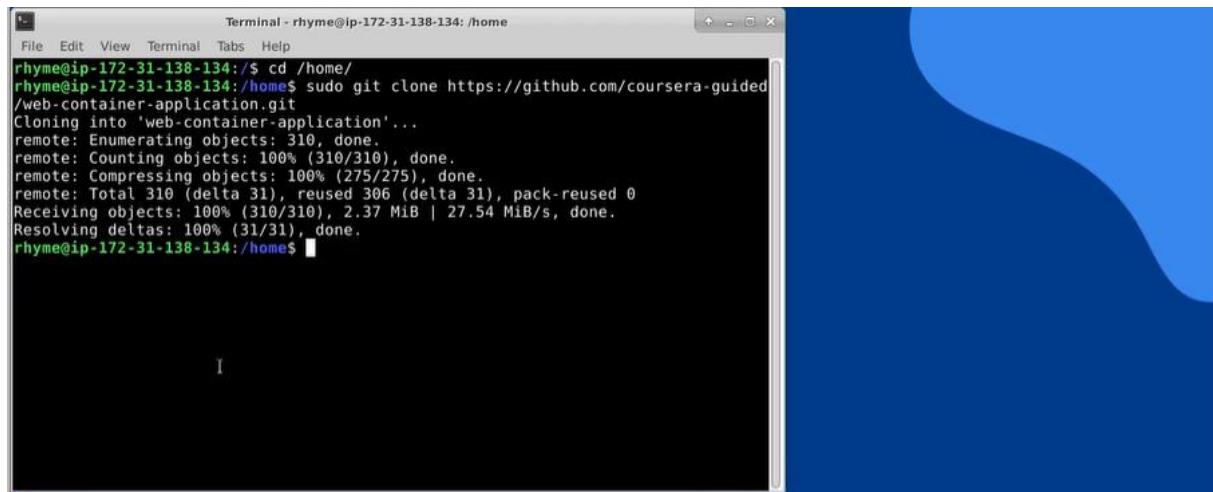
Create repository

Repository name	URI	Created at	Tag immutability	Scan on push	Encryption type
website	827905500843.dkr.ecr.us-east-1.amazonaws.com/website	Jun 21, 2021 11:00:04 PM	Disabled	Disabled	AES-256



Task 4: Clone an application from GitHub, build a Docker image, run a container and push the image to Elastic Container Registry

We need to create a container image for our web application, we will clone the application files and GitHub in the cloud desktop



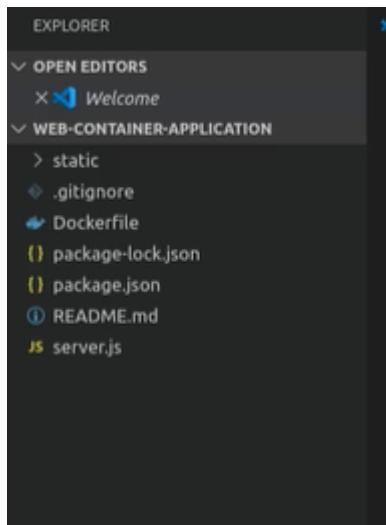
A screenshot of a terminal window titled "Terminal - rhyme@ip-172-31-138-134: /home". The window shows the command "sudo git clone https://github.com/coursera-guided/web-container-application.git" being run. The output of the command is displayed, showing the progress of cloning the repository, including object enumeration, counting, compressing, receiving objects, and resolving deltas. The process is completed successfully.

And on typing ls command, we can see that we have a web dash container dash application:



A screenshot of a terminal window titled "Terminal - rhyme@ip-172-31-138-134: /home\$ ls". The command "ls" is run, and the output shows three files: "ubuntu", "web-container-application", and ".gitignore".

To explore the application, we gonna open it with an IDE, and then we see that It's a NodeJS JavaScript application



The application has a Docker file, it is the manifest file to create and build our container image. Let's see its content:



```
Dockerfile - web-container-application - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER Dockerfile x
OPEN EDITORS Dockerfile
WEB-CONTAINER-APPLICATION
static
.gitignore
Dockerfile
package-lock.json
package.json
README.md
server.js
Dockerfile x
FROM alpine:3.13 AS baseimage
RUN apk --no-cache add ca-certificates musl-dev wget bash curl
RUN apk add ca-certificates && update-ca-certificates
RUN apk add --update nodejs npm
RUN apk add --update tzdata
RUN mkdir /web
COPY . /web
WORKDIR /web
RUN npm install
EXPOSE 8089
CMD [ "node", "server.js" ]
```

It's a very simple image where NodeJS is installed in our application is exposed on the port 8089.

Now, let's build our image:

```
#sudo docker build -f Dockerfile -t website:latest .
```

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL 1:sudo
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker build -f Dockerfile -t website:latest .
Sending build context to Docker daemon 7.528MB
Step 1/11 : FROM alpine:3.13 AS baseimage
3.13: Pulling from library/alpine
540db60ca938: Pull complete
Digest: sha256:f51ff2d96627690d62fee79e6eecd9fa87429a38142b5df8a3bfbb26061df7fc
Status: Downloaded newer image for alpine:3.13
--> 6dbb9cc54074
Step 2/11 : RUN apk --no-cache add ca-certificates musl-dev wget bash curl
--> Running in 323897bead16
```

And to see our list of images, we can type:

```
#sudo docker images
```

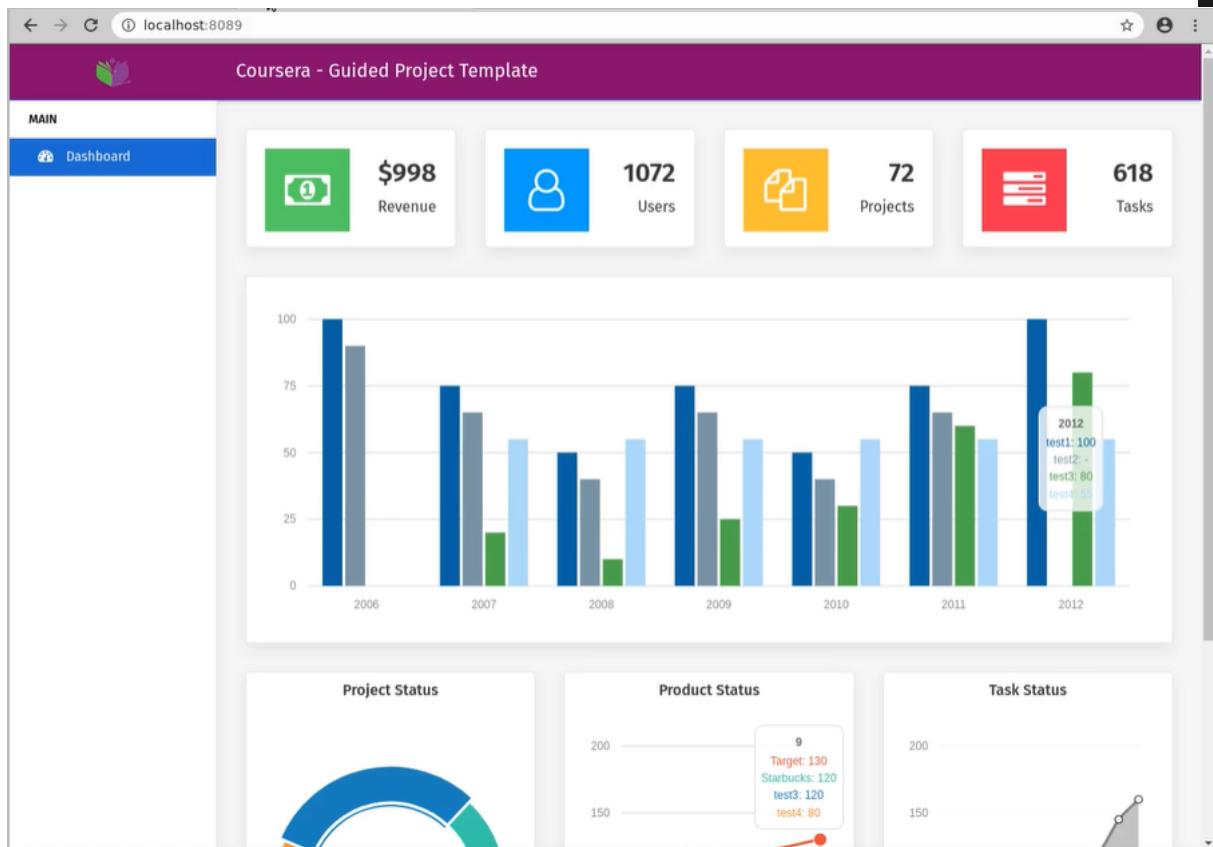
```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
website latest 873935ebf736 11 seconds ago 90.3MB
alpine 3.13 6dbb9cc54074 2 months ago 5.61MB
```

Then now, to run and validate image, we gonna use the command:

```
#sudo docker run -p 8089:8089 website:latest
```

```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker run -p 8089:8089 website:latest
Your app is listening on port 8089
```

Now we can see our application on the browser:



On typing the command: #sudo docker ps, we can see that the container is running:

```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS N
AMES
8528bc57f20c website:latest "node server.js" 53 seconds ago Up 51 seconds 0.0.0.0:8089->8089/tcp c
```

To stop the container, we use the command:

```
#sudo docker stop
```

```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker stop 8528bc57f20c
8528bc57f20c
```

And we can check with:

```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

Like we can see, we don't have any container running.

Now let's push our image to our elastic container registry. Pushing the image to the last container registry repository allow us to use this image in the last Kubernetes service.

We go to the Elastic container registry repositories:



Search results for 'ecr'

Services (6)

Features (4)

Documentation (30,978)

Marketplace (12)

Services

See all 6 results ▾

Secrets Manager

Easily rotate, manage, and retrieve secrets throughout their lifecycle

Elastic Container Service

Highly secure, reliable, and scalable way to run containers

Elastic Container Registry

Fully-managed Docker container registry : Share and deploy container software, publ...

Top features

Repositories Registries

Key Management Service

Later, we select our project called 'website'

Amazon Container Services

ECR > Repositories

Private Public

Private repositories (1 of 1)

Find repositories

Repository name	URI	Created at	Tag immutability	Scan on push	Encryption type
website	827905500843.dkr.ecr.us-east-1.amazonaws.com/website	Jun 21, 2021 11:00:04 PM	Disabled	Disabled	AES-256

And we click on 'View push commands':

Amazon Container Services

ECR > Repositories

Private Public

Private repositories (1 of 1)

Find repositories

Repository name	URI	Created at	Tag immutability	Scan on push	Encryption type
website	827905500843.dkr.ecr.us-east-1.amazonaws.com/website	Jun 21, 2021 11:00:04 PM	Disabled	Disabled	AES-256

Then, we copy the first command in this page:



Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin
```
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t website .
```
3. After the build completes, tag your image so you can push the image to this repository:

And now we paste it in the terminal of our IDE, with adding the term ‘sudo’ before the docker command:

```
rhyme@ip-172-31-138-134:/home/web-container-application$ aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin 827905500843.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/rhyme/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

We don't need to execute the second command because we are ready that not a few steps behind, we gonna copy the third command and paste it, without forgot to prefix the docker command with ‘sudo’:

```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker tag website:latest 827905500843.dkr.ecr.us-east-1.amazonaws.com/website:latest
rhyme@ip-172-31-138-134:/home/web-container-application$
```

We can see we have our images in the new tech for our image.

```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker images
REPOSITORY          TAG      IMAGE ID   CREATED     SIZE
827905500843.dkr.ecr.us-east-1.amazonaws.com/website    latest    873935ebf736  5 minutes ago  90.3MB
website             latest    873935ebf736  5 minutes ago  90.3MB
alpine              3.13     6dbb9cc54074  2 months ago  5.61MB
```

And finally we gonna a copy the latest command:

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 827905500843.dkr.ecr.us-east-1.amazonaws.com/website:latest
```



```
rhyme@ip-172-31-138-134:/home/web-container-application$ sudo docker push 827905500843.dkr.ecr.us-east-1.amazonaws.com/website:latest
The push refers to repository [827905500843.dkr.ecr.us-east-1.amazonaws.com/website]
73237bfd4927: Pushed
d3b1a41dc061: Pushed
8b0de1cb6102: Pushed
d788eef9a8c7: Pushed
fb525caa4887: Pushed
219d41de9e7e: Pushed
7a94da318d8a: Pushed
b2d5eeeaba3a: Pushed
latest: digest: sha256:0343a96f66b05864a6149016c61e4031d3118dbcfe1c82e941b7f2b135c03a0c size: 2001
```

Then our image will be pushed to the ECR repository. And like we see our image is here:

The screenshot shows the AWS ECR service interface. On the left, there's a sidebar with 'Amazon Container Services' selected, followed by 'Amazon ECS', 'Amazon EKS', and 'Amazon ECR'. Under 'Amazon ECR', 'Images' is highlighted. The main area shows a 'website' repository with one image tag listed: 'latest'. The table includes columns for Image tag, Pushed at, Size (MB), Image URI, Digest, Scan status, and Vulnerability. The 'latest' tag was pushed on Jun 21, 2021, at 11:12:50 PM, with a size of 36.86 MB and a digest of sha256:0343a96...

Task 5: Create an Elastic Kubernetes Cluster and set all configuration needed

We have the AWS console opened, we will to create our Kubernetes cluster on AWS. So, firstly we will to search for EKS and we will choose Elastic Kubernetes Service:

The screenshot shows the AWS search results for 'eks'. In the search bar, 'eks' is typed. Below the search bar, it says 'Search results for "eks"' and 'Services (1)'. There is one result: 'Elastic Kubernetes Service' with the sub-description 'The most trusted way to run Kubernetes'. Below this, there are sections for 'Features' and 'Clusters'.

Later, in the left menu below Amazon EKS we click on Clusters:



Serviços ▾ Search for services, features, marketplace products, and docs [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support ▾

Amazon Container Services

Amazon ECS Clusters Task definitions

Amazon EKS Clusters

Amazon ECR Repositories

Containers

Elastic Kubernetes Service (Amazon EKS)

Fully managed Kubernetes control plane

Amazon EKS is a managed service that makes it easy for you to use Kubernetes on AWS without needing to install and operate your own Kubernetes control plane.

Create EKS cluster

Cluster name Next step

Pricing

EKS Control Plane EKS Pricing

How it works

To create a cluster, we should define a name for the cluster, the Kubernetes version and the cluster role:

Services ▾ Search for services, features, marketplace products, and docs [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support ▾

EKS > Clusters > Create EKS cluster

Step 1 Configure cluster

Step 2 Specify networking

Step 3 Configure logging

Step 4 Review and create

Configure cluster

Cluster configuration Info

Name - Not editable after creation.
Enter a unique name for this cluster.

Kubernetes version Info
Select the Kubernetes version for this cluster.

Cluster Service Role Info - Not editable after creation.
Select the IAM Role to allow the Kubernetes control plane to manage AWS resources on your behalf.
To create a new role, go to the [IAM console](#).

Secrets encryption Info
Once enabled, secrets encryption cannot be modified or removed.

Enable envelope encryption of Kubernetes secrets using KMS
Enable envelope encryption to provide an additional layer of encryption for your Kubernetes secrets.

Tags (0) Info



EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Review and create

Specify networking

Networking Info
These properties cannot be changed after the cluster is created.

VPC Info
Select a VPC to use for your EKS Cluster resources.
To create a new VPC, go to the [VPC console](#).

vpc-5752ab2a | Default

Subnets Info
Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster.
To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

Filter subnets

<input type="checkbox"/> subnet-cf7e18c1 us-east-1f 172.31.64.0/20	worker node subnets.
<input checked="" type="checkbox"/> subnet-9ee844f8 us-east-1b 172.31.0.0/20	<input type="button" value="Remove"/>
<input type="checkbox"/> subnet-eb71daca us-east-1c 172.31.80.0/20	<input type="button" value="Remove"/>
<input checked="" type="checkbox"/> subnet-668d2439 us-east-1a 172.31.32.0/20	<input type="button" value="Remove"/>
<input type="checkbox"/> subnet-210dfd10 us-east-1e 172.31.48.0/20	<input type="button" value="Remove"/>
<input type="checkbox"/> subnet-2bbfc366 us-east-1d 172.31.16.0/20	<input type="button" value="Remove"/>

Configure access to the Kubernetes API server endpoint.

And we select the public cluster endpoint, and we keep the other configurations as is:

Step 4
Review and create

vpc-5752ab2a | Default

Subnets Info
Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster.
To create a new subnet, go to the [VPC console](#).

Select subnets

subnet-9ee844f8 subnet-668d2439

Security groups Info
Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.
To create a new security group, go to the corresponding page in the [VPC console](#).

Select security groups

sg-99204da5

Configure Kubernetes Service IP address range Info
Specify the range from which cluster services will receive IP addresses.

Cluster endpoint access Info
Configure access to the Kubernetes API server endpoint.

Public Info
The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

Public and private
The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

Private
The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

, and we click -> **Next**:



aws Services ▾ Search for services, features, marketplace products, and docs [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support

EKS > Clusters > Create EKS cluster

Step 1 Configure cluster

Step 2 Specify networking

Step 3 Configure logging

Step 4 Review and create

Configure logging

Control Plane Logging Info

CloudWatch log group
Send audit and diagnostic logs from the Amazon EKS control plane to CloudWatch Logs.

API server
Logs pertaining to API requests to the cluster.
 Disabled

Audit
Logs pertaining to cluster access via the Kubernetes API.
 Disabled

Authenticator
Logs pertaining to authentication requests into the cluster.
 Disabled

Controller manager
Logs pertaining to state of cluster controllers.
 Disabled

Scheduler
Logs pertaining to scheduling decisions.
 Disabled

Cancel Previous Next

And -> **Next**

aws Services ▾ Search for services, features, marketplace products, and docs [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support

vpc-5752ab2a	subnet-9ee844f8 subnet-668d2439	sg-99204da5
--------------	------------------------------------	-------------

Cluster endpoint access

API server endpoint access
Public

Networking add-ons

Amazon VPC CNI Version v1.7.5-eksbuild.2	CoreDNS Version v1.8.3-eksbuild.1	kube-proxy Version v1.20.4-eksbuild.2
--	---	---

Step 3: Configure logging Edit

Control Plane Logging

API server Disabled	Audit Disabled	Authenticator Disabled
Controller manager Disabled	Scheduler Disabled	

Cancel Previous Create

And -> **Create**



The screenshot shows the AWS EKS Cluster configuration page. At the top, a blue banner says "Cluster creation in progress" and "coursera_cluster is now being created. This process may take several minutes." Below this, the cluster name "coursera_cluster" is displayed with a status of "Creating". The "Configuration" tab is selected. In the "Cluster configuration" section, it shows Kubernetes version 1.20 and Platform version eks.1. Below this, the "Details" tab is selected, showing the API server endpoint and Cluster ARN (arn:aws:eks:us-east-1:827905500843:cluster/coursera_cluster).

The cluster will be created and it will take several minutes, generally 15-20minutes

Task 6: Create and set a Node Group to host containers in a POD architecture

In this task we will create our load group which is the group of virtual machines that will be part of our cluster and where our containers will run.

Firstly, we gonna check if our clusters was created and if is active. In the AWS console, we go to the search bar on to top and we type EKS and we click to Elastic Kubernetes Service:

The screenshot shows the AWS search results for 'eks'. The search bar at the top has 'eks' typed into it. The results are categorized into Services (1), Features (1), Documentation (30,273), and Marketplace (226). The 'Elastic Kubernetes Service' result is highlighted, showing its description: "The most trusted way to run Kubernetes". Below this, there are sections for 'Clusters' (Elastic Kubernetes Service feature) and 'Documentation' (Amazon EKS networking - Amazon EKS User Guide).

And click on -> Clusters



The screenshot shows the AWS EKS Clusters page. On the left, there's a sidebar for 'Amazon Container Services' with options for Amazon ECS (Clusters, Task definitions), Amazon EKS (Clusters), and Amazon ECR (Repositories). The main area shows a table titled 'Clusters (1)'. It has three columns: 'Cluster name', 'Kubernetes version', and 'Status'. A single row is listed with 'coursera_cluster' under 'Cluster name', '1.20' under 'Kubernetes version', and 'Active' with a green checkmark under 'Status'. There are buttons for 'Edit', 'Delete', and 'Create cluster' at the top right of the table.

In this page above, we see the status of the cluster which we created in the previous task is active, it means our cluster is running. So, we click on the cluster name to open the cluster:

The screenshot shows the 'Configure Node Group' step in the AWS EKS wizard. The left sidebar lists steps: Step 1 (Configure Node Group), Step 2 (Set compute and scaling configuration), Step 3 (Specify networking), and Step 4 (Review and create). The main area is titled 'Configure Node Group' and contains a section for 'Node Group configuration'. It says: 'A Node Group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple Node Groups to your cluster.' Below this is a 'Name' field containing 'coursera_nodegroup'. Under 'Node IAM Role', a dropdown menu is set to 'courseranode_role'. A tooltip message states: 'The selected role must not be used by a self-managed node group as this could lead to a service interruption upon Managed Node Group deletion.' At the bottom, there's a 'Launch template' section with a radio button for 'Use launch template'.

And we click on -> **Next**

And we fill the configuration:



EKS > Clusters > coursera_cluster > Add Node Group

Step 1
Configure Node Group

Step 2
Set compute and scaling configuration

Step 3
Specify networking

Step 4
Review and create

Set compute and scaling configuration

Node Group compute configuration

These properties cannot be changed after the Node Group is created.

AMI type [Info](#)
Select the EKS-optimized Amazon Machine Image for nodes.
Amazon Linux 2 (AL2_x86_64)

Capacity type
Select the capacity purchase option for this Node Group.
On-Demand

Instance types [Info](#)
Select instance types you prefer for this Node Group.
Select

t3.medium X
vCPU: Up to 2 vCPUs memory: 4.0 GiB

Disk size
Select the size of the attached EBS volume for each node.
20 GiB

Node Group scaling configuration

Minimum size
Set the minimum number of nodes that the group can scale in to.
2 nodes

Maximum size
Set the maximum number of nodes that the group can scale out to.
4 nodes

Maximum node size must be greater than or equal to 1 and cannot be lower than the minimum size

Desired size
Set the desired number of nodes that the group should launch with initially.
2 nodes

Desired node size must be greater than or equal to 0

Cancel Previous **Next**

A screenshot of the AWS EKS Node Group configuration wizard. It shows the 'Compute and Scaling Configuration' step. Under 'Compute Configuration', the AMI type is set to 'Amazon Linux 2 (AL2_x86_64)', capacity type to 'On-Demand', and an instance type 't3.medium' is selected. The 'Scaling Configuration' section shows a minimum size of 2, a maximum size of 4, and a desired size of 2. The 'Next' button is highlighted in orange at the bottom right.

And we click on -> **Next**



aws Services ▾ Search for services, features, marketplace products, and docs [Alt+S] Coursera Universidade Global ▾ N. Virginia ▾ Support ▾

EKS > Clusters > coursera_cluster > Add Node Group

Step 1 Configure Node Group

Step 2 Set compute and scaling configuration

Step 3 Specify networking

Step 4 Review and create

Review and create

Step 1: Configure Node Group

Edit

Node Group configuration

Name: coursera_nodegroup

Node IAM Role: arn:aws:iam::827905500843:role/courseranode_role

Kubernetes labels (0)

Filter by key or value < 1 >

Key	Value
No labels	

This Node Group does not have any Kubernetes labels.

Kubernetes taints (0)

Filter by key, value or effect < 1 >

Key	Value	Effect
No taints		

Tags (0)

Filter by key or value < 1 >

Key	Value
No tags	

This Node Group does not have any tags.

Step 2: Set compute and scaling configuration

Edit

Node Group compute configuration

Capacity type On-Demand	Instance types t3.medium	Disk size 20
AMI type Amazon Linux 2 (AL2_x86_64)		

Node Group scaling configuration

Minimum size 2 nodes	Maximum size 4 nodes	Desired size 2 nodes
-------------------------	-------------------------	-------------------------

Step 3: Specify networking

Edit

Node Group network configuration

Subnets subnet-9ee844f8 subnet-668d2439	Allow remote access to nodes Enabled SSH key pair coursera_keypair	Allow remote access from All
---	---	---------------------------------

Cancel Previous Create



And later, we click on -> **Create**

Then, the virtual machines and his sources will be created.

Task 7: Deploy an application in Elastic Kubernetes Cluster using an Image hosted in Elastic Container Registry

In this task we will deploy the web application we have built in task four and push it to Elastic Container Registry into our Elastic Kubernetes Cluster. To do that we will use a deployment

Let's first set the cube conflict credentials to connect in our cluster by **kubectl**

```
rhyme@ip-172-31-138-134:/$ aws eks --region us-east-1 update-kubeconfig --name coursera_cluster
Added new context arn:aws:eks:us-east-1:827905500843:cluster/coursera_cluster to
/home/rhyme/.kube/config
rhyme@ip-172-31-138-134:/$
```

We can see if the credentials are working by typing **kubectl**:

```
rhyme@ip-172-31-138-134:/$ kubectl get nodes
NAME           STATUS    ROLES      AGE   VERSION
ip-172-31-13-155.ec2.internal   Ready     <none>    7m26s   v1.20.4-eks-6b7464
ip-172-31-38-179.ec2.internal   Ready     <none>    7m20s   v1.20.4-eks-6b7464
```

We can see our two notes are listed here.



```
! deployment.yaml ×  
home > rhyme > Desktop > code > ! deployment.yaml  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: website  
5    labels:  
6      app: website  
7  spec:  
8    replicas: 5  
9    selector:  
10      matchLabels:  
11        app: website  
12    template:  
13      metadata:  
14        labels:  
15          app: website  
16      spec:  
17        containers:  
18          - name: website  
19            image: <your image here>  
20            ports:  
21              - containerPort: 8089  
22            resources:  
23              requests:  
24                cpu: 500m  
25                memory: 256Mi  
26              limits:  
27                cpu: 500m  
28                memory: 256Mi  
29  ---  
30  apiVersion: v1  
31  kind: Service  
32  metadata:  
33    name: website  
34  spec:  
35    selector:  
36      app: website  
37    ports:  
38      - port: 8089  
39        targetPort: 8089  
40    type: LoadBalancer
```

This deployment file has two parts, the first one is deployments. Deployments will deploy our pods, our containers. So, we need to define how many replicas, it means how many containers we want to deploy. In this example we gonna define 5 replicas. And the name of our application will be ‘website’.

Now, we gone to the Elastic Container Registry on the AWS console, and we click on our website repository:



The screenshot shows the AWS ECR (Amazon Container Registry) interface. On the left, there's a sidebar for 'Amazon Container Services' with sections for Amazon ECS, Amazon EKS, and Amazon ECR. Under ECR, 'Repositories' is selected. The main area shows a table titled 'Private repositories (1)'. One repository is listed: 'website' (URI: 827905500843.dkr.ecr.us-east-1.amazonaws.com/website), created on Jun 21, 2021, at 11:00:04 PM. It has 'Disabled' status for both 'Tag immutability' and 'Scan on push', and 'AES-256' for 'Encryption type'. There are buttons for 'View push commands', 'Delete', 'Edit', and 'Create repository'.

And, we copy the URI for the latest version:

This screenshot shows the same ECR interface, but now focusing on the 'website' repository. In the 'Images (1)' section, the 'latest' image is listed with a push date of Jun 21, 2021, at 11:12:50 PM. The size is 36.86 MB. A 'Copy URI' button is visible next to the image details. A tooltip 'Image URI copied' appears over the 'Copy' button. There are also 'Delete' and 'Scan' buttons.

And we paste the URI in the image session of the deployment file:



```
# deployment.yaml •  
home > rhyme > Desktop > code > ! deployment.yaml  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: website  
5    labels:  
6      app: website  
7  spec:  
8    replicas: 5  
9    selector:  
10      matchLabels:  
11        app: website  
12    template:  
13      metadata:  
14        labels:  
15          app: website  
16    spec:  
17      containers:  
18        - name: website  
19          image: |  
20            ports:  
21              - containerPort: 80  
22            resources:  
23              requests:  
24                cpu: 500  
25                memory: 1Gi  
26              limits:  
27                cpu: 500  
28                memory: 256Mi  
29    ...  
30  apiVersion: v1  
31  kind: Service  
32  metadata:  
33    name: website
```



A context menu is open over the 'memory:' field in line 25 of the YAML file. The menu items are: Change All Occurrences (Ctrl+F2), Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V, highlighted with a blue selection bar), and Command Palette... (Ctrl+Shift+P).



```
# deployment.yaml •  
home > rhyme > Desktop > code > # deployment.yaml  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: website  
5    labels:  
6      app: website  
7  spec:  
8    replicas: 5  
9    selector:  
10      matchLabels:  
11        app: website  
12      template:  
13        metadata:  
14          labels:  
15            app: website  
16        spec:  
17          containers:  
18            - name: website  
19              image: 827905500843.dkr.ecr.us-east-1.amazonaws.com/website:latest  
20            ports:  
21              - containerPort: 8089  
22            resources:  
23              requests:  
24                cpu: 500m  
25                memory: 256Mi  
26              limits:  
27                cpu: 500m  
28                memory: 256Mi  
29    ...  
30  apiVersion: v1  
31  kind: Service  
32  metadata:
```

The next part of the deployment file is servicing:

```
apiVersion: v1  
kind: Service  
metadata:  
  name: website  
spec:  
  selector:  
    app: website  
  ports:  
    - port: 8089  
      targetPort: 8089  
  type: LoadBalancer
```

Where we define services for our application like: load balancer



Now, we will save the file, and go to the terminal in our IDE, and type the commands:

```
rhyume@ip-172-31-138-134:~/Desktop/code$ cd Desktop/code
rhyume@ip-172-31-138-134:~/Desktop/code$ kubectl apply -f deployment.yaml
deployment.apps/website created
service/website created
```

Like we see our deployment and service will be created in our cluster.

To verify our elements, we can continue using the **kubectl** to list the deployments:

```
rhyume@ip-172-31-138-134:~/Desktop/code$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
Website   5/5     5           5           23s
```

Then, we can see our deployment to list the pods we can use:

```
rhyume@ip-172-31-138-134:~/Desktop/code$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
website-77f9f8c9cc-598b4   1/1    Running   0          37s
website-77f9f8c9cc-8q8cx   1/1    Running   0          37s
website-77f9f8c9cc-9sjgm   1/1    Running   0          37s
website-77f9f8c9cc-cmk99   1/1    Running   0          37s
website-77f9f8c9cc-slfrm   1/1    Running   0          37s
```

And like we see this is the pods are being deploying and now is running.

Now, if we want to see more details about or pods, we can use this command: **#kubectl get pods -o wide**

```
rhyume@ip-172-31-138-134:~/Desktop/code$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATE
S
website-77f9f8c9cc-598b4  1/1    Running   0          55s  172.31.11.147  ip-172-31-13-155.ec2.internal  <none>  <none>
website-77f9f8c9cc-8q8cx  1/1    Running   0          55s  172.31.12.4   ip-172-31-13-155.ec2.internal  <none>  <none>
website-77f9f8c9cc-9sjgm  1/1    Running   0          55s  172.31.14.41  ip-172-31-13-155.ec2.internal  <none>  <none>
website-77f9f8c9cc-cmk99  1/1    Running   0          55s  172.31.38.241 ip-172-31-38-179.ec2.internal  <none>  <none>
website-77f9f8c9cc-slfrm  1/1    Running   0          55s  172.31.47.170 ip-172-31-38-179.ec2.internal  <none>  <none>
```

So, we can see for example, which node, each pod was deployed in the IP for each pod.

We can also see our nodes by: **#kubectl get nodes**

```
rhyume@ip-172-31-138-134:~/Desktop/code$ kubectl get nodes
NAME           STATUS   ROLES   AGE   VERSION
ip-172-31-13-155.ec2.internal  Ready   <none>   17m   v1.20.4-eks-6b7464
ip-172-31-38-179.ec2.internal  Ready   <none>   17m   v1.20.4-eks-6b7464
```

Similarly, we can have a list of our services:

```
rhyume@ip-172-31-138-134:~/Desktop/code$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.100.0.1   <none>        443/TCP     28m
website    LoadBalancer  10.100.239.188 ac7038f3a11434aa3a6544d4b9538b48-1318058674.us-east-1.elb.amazonaws.com  8089:30638/TCP  84s
```

Now, before access, let's go to our load balancer security group configuration and make our application available for any source.



So, on going to the AWS console, in the left menu, let's click on '**Load balancer**:

The screenshot shows the AWS Lambda service page. On the left, there is a navigation sidebar with various AWS services listed under 'Services'. Under the 'Load Balancing' section, 'Load Balancers' is selected. The main content area displays a table of load balancers. One row is selected, showing details for a load balancer named 'ac7038f3a11434aa3a6544d4b9538b48'. The table includes columns for Name, DNS name, State, VPC ID, and Availability Zones. The 'Availability Zones' column shows 'us-east-1f, us-east-1e, ...'. Below the table, there is a detailed view of the selected load balancer, including sections for 'Basic Configuration', 'Port Configuration', 'Security', and 'Attributes'.

Later, we scroll down, and click on this link:

The screenshot shows the AWS Lambda service page. On the left, there is a navigation sidebar with various AWS services listed under 'Services'. Under the 'Load Balancing' section, 'Load Balancers' is selected. The main content area displays a table of load balancers. One row is selected, showing details for a load balancer named 'ac7038f3a11434aa3a6544d4b9538b48'. The table includes columns for Name, DNS name, State, VPC ID, and Availability Zones. The 'Availability Zones' column shows 'us-east-1f, us-east-1e, ...'. Below the table, there is a detailed view of the selected load balancer, including sections for 'Basic Configuration', 'Port Configuration', 'Security', and 'Attributes'.



Security Groups acts like a firewall to control the inbound and outbound in our load balancer. We check if we have in the inbound rule open for our peace.

Name	Security group ID	Security group name	VPC ID	Description
eks-cluster-sg-cour...	sg-03bc1213c6ee3aa01	eks-cluster-sg-courser...	vpc-5752ab2a	EKS created security group
-	sg-09a7db52020ad10b8	k8s-elb-ac7038f3a114...	vpc-5752ab2a	Security group for Kubernetes ELB

Like we see, we have 0.000/0.

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	8089	0.0.0.0/0	-
Custom ICMP - IPv4	Destination Unreachable	fragmentation required, and DF flag set	0.0.0.0/0	-

Let's delete any other rule:



sg-09a7db52020ad10b8 - k8s-elb-ac7038f3a11434aa3a6544d4b9538b48

Actions ▾

Details

Security group name

sg-09a7db52020ad10b8 - k8s-elb-ac7038f3a11434aa3a6544d4b9538b48

Security group ID

sg-09a7db52020ad10b8

Description

Security group for Kubernetes ELB
ac7038f3a11434aa3a6544d4b9538b48 (default/website)

VPC ID

vpc-5752ab2a

Owner

827905500843

Inbound rules count

2 Permission entries

Outbound rules count

1 Permission entry

Inbound rules

Outbound rules

Tags

Inbound rules (2)

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	8089	0.0.0.0/0	-
Custom ICMP - IPv4	Destination Unreachable	fragmentation required, and DF flag set	0.0.0.0/0	-

Click on → Edit inbound rules

EC2 > Security Groups > sg-09a7db52020ad10b8 - k8s-elb-ac7038f3a11434aa3a6544d4b9538b48 > Edit inbound rules

Edit inbound rules

Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Info

Type	Protocol	Port range	Source	Description - optional	Info
Custom TCP	TCP	8089	Custom	Q	<input type="text"/> 0.0.0.0/0 X
Custom ICMP - IPv4	Destina...	fragmentation re...	Custom	Q	<input type="text"/> 0.0.0.0/0 X

Add rule

⚠ NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel

Preview changes

Save rules



Screenshot of the AWS EC2 Security Groups 'Edit inbound rules' page. The interface shows a table with columns: Type, Protocol, Port range, Source, and Description - optional. A single rule is listed: Custom TCP on port 8089 from 0.0.0.0/0. A note at the bottom states: '⚠ NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.' Buttons at the bottom include 'Cancel', 'Preview changes', and 'Save rule'.

We Clicked on -> Delete -> Save rules

Later, we go back on the load balancer configuration page and we copy the DNS name:

Screenshot of the AWS Load Balancers 'Create Load Balancer' page. The left sidebar shows navigation options like Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images, Elastic Block Store, Network & Security, and Load Balancing. In the main area, a table lists a single load balancer entry: ac7038f3a11434aa3a6544d4b9538b48. Below it, the 'Basic Configuration' section shows details: Name (ac7038f3a11434aa3a6544d4b9538b48), Creation time (June 21, 2021 at 11:56:14 PM UTC), Hosted zone (Z35SXDOTRQ7X7K), DNS name (ac7038f3a11434aa3a6544d4b9538b48-1318058674.us-east-1), Type (Classic (Migrate Now)), Scheme (internet-facing), and Availability Zones (subnet-210df10 - us-east-1e, subnet-2bbfc366 - us-east-1d, subnet-668d2439 - us-east-1a, subnet-9ee844f8 - us-east-1b, subnet-cf7e18c1 - us-east-1f, subnet-71f1aca - us-east-1c). A context menu is open over the DNS name field, with 'Copy' selected.

Now, we pass to a new tab and complete the URL with the port 8089, which is the port our container application is using:



And we paste it:

The screenshot shows a web browser with three tabs open. The active tab displays a Google search result for the URL ac7038f3a11434aa3a6544d4b9538b48-1318058674.us-east-1.elb.amazonaws.com:8089. The search results page for 'Coursera - Guided Project Template' is shown, featuring a bar chart and three donut charts.

Google Search Results:

- 1. ac7038f3a11434aa3a6544d4b9538b48-1318058674.us-east-1.elb.amazonaws.com:8089
- 2. ac7038f3a11434aa3a6544d4b9538b48-1318058674.us-east-1.elb.amazonaws.com:8089 - Google Search

Coursera - Guided Project Template Dashboard:

- MAIN:** Dashboard
- Key Metrics:**
 - Revenue: \$998
 - Users: 1072
 - Projects: 72
 - Tasks: 618
- Bar Chart:** Shows revenue over time from 2006 to 2012. A tooltip for 2012 indicates values for test1 (100), test2 (~120), test3 (80), and test4 (60).
- Project Status:** Donut chart showing project completion status.
- Product Status:** Donut chart showing product status.
- Task Status:** Line chart showing task progress over time.

Like we see, our population is running in our EKS cluster on AWS.

We can also open our EKS cluster dashboard to see the workload steps, to see our application deployment



Elastic Kubernetes Service -> Clusters -> Coursera_cluster

The screenshot shows the AWS EKS Clusters page for the 'coursera_cluster'. The left sidebar lists 'Amazon Container Services', 'Amazon ECS' (Clusters, Task definitions), 'Amazon EKS' (Clusters), and 'Amazon ECR' (Repositories). The main content area shows the 'coursera_cluster' details, including a message about new add-on versions. The 'Workloads' tab is selected, displaying four entries: 'website' (Deployment, 5 pods, 5 Ready | 0 Failed | 5 Desired), 'coredns' (Deployment, 2 pods, 2 Ready | 0 Failed | 2 Desired), 'aws-node' (DaemonSet, 2 pods, 2 Ready | 0 Failed | 2 Desired), and 'kube-node' (DaemonSet, 2 pods, 2 Ready | 0 Failed | 2 Desired).

Name	Namespace	Type	Created	Last transition time	Pod count	Status
website	default	Deployment	5 minutes ago	5 minutes ago	5	5 Ready 0 Failed 5 Desired
coredns	kube-system	Deployment	32 minutes ago	21 minutes ago	2	2 Ready 0 Failed 2 Desired
aws-node	kube-system	DaemonSet	32 minutes ago	-	2	2 Ready 0 Failed 2 Desired
kube-node	kube-system	DaemonSet	32 minutes ago	-	2	2 Ready 0 Failed 2 Desired

Then like we see in the picture above, our deployment is running in our cluster.

Task 8: Scale the Node Group to support the increase of new pods in the Kubernetes cluster

In this task, we gonna see how we can scale our cluster and deploy more containers for our application based on the environment we have deployed. Our instance can support around 17 pods in total.

Considering we have application pods and Kubernetes system pods, it means if we need more pods, more containers for our application, we need to scale our load group at the more virtual machines.

We can do that manually, but the big advantage of using a management services like Elastic Kubernetes services is to make it automatically.

Firstly, we gone to AWS console and see our cluster and node group configuration:

Elastic Kubernetes Service -> Clusters -> Coursera_cluster -> configurations



Screenshot of the AWS EKS Cluster configuration page for 'coursera_cluster'.

The cluster is active and has 1 add-on available. The Configuration tab is selected.

Cluster configuration details:

- Kubernetes version: 1.20
- Platform version: eks.1

Details section:

API server endpoint: https://9338796F0C9618BCD820F609D3C65821.eks.amazonaws.com	OpenID Connect provider URL: https://oidc.eks.us-east-1.amazonaws.com/id/9338796F0C9618BCD820F609D3C65821	Cluster ARN: arn:aws:eks:us-east-1:827905500843:cluster/coursera_cluster
Creation time: Jun 21st 2021 at 11:38 PM	Certificate authority	Cluster IAM Role ARN

-> compute

Screenshot of the AWS EKS Node Group configuration page for the 'coursera_cluster' node group.

The node group is active and has 1 node.

Node Group configuration details:

Kubernetes version: 1.20	AMI type: AL2_x86_64	Status: Active
AMI release version: 1.20.4-20210526	Instance types: t3.medium	Disk size: 20 GiB

Details section:

Node Group ARN: arn:aws:eks:us-east-1:827905500843:nodegroup/coursera_nodegroup/84bd1878-6d74-c5ed-83bb-caaa5bc032d1	Autoscaling group name: eks-84bd1878-6d74-c5ed-83bb-caaa5bc032d1	Capacity type: On-Demand	Subnets: subnet-9ee844fb, subnet-668d2459
Creation time: Jun 21st 2021 at 11:38 PM	Node IAM Role ARN: arn:aws:iam::827905500843:role/courseranode_role	Minimum size: 2 nodes	Allow remote access to nodes: Enabled
		Maximum size: 4 nodes	SSH key pair: coursera_keypair
		Desired size: 2 nodes	Allow remote access from: All

We see in capacity type we have two nodes minimal for nodes maximum and two nodes desire it.



Now, we click on nodes Notes step

The screenshot shows the AWS Lambda console interface. The top navigation bar includes 'Details', 'Nodes' (which is highlighted in orange), 'Health issues (0)', 'Kubernetes labels', 'Kubernetes taints', 'Update history', and 'Tags'. Below this, a section titled 'Nodes (2)' contains a search bar and a table with the following data:

Node name	Instance type	Node Group	Created	Status
ip-172-31-13-155.ec2.internal	t3.medium	coursera_nodegroup	32 minutes ago	Ready
ip-172-31-38-179.ec2.internal	t3.medium	coursera_nodegroup	32 minutes ago	Ready

And like we see, we have two nodes up and running.

And if we go to our closer workload information, we see we have our application deployment running five boards.

The screenshot shows the AWS Lambda console interface with the 'Workloads' tab selected (highlighted in orange). The top navigation bar includes 'Overview', 'Workloads' (highlighted in orange), and 'Configuration'. Below this, a section titled 'Workloads (4)' contains a search bar and a table with the following data:

Name	Namespace	Type	Created	Last transition time	Pod count	Status
website	default	Deployment	16 minutes ago	16 minutes ago	5	5 Ready 0 Failed 5 Desired

It means five containers.

Kubernetes can be installed in our cluster using a deployment like we did for our application.

We have an E M O file prepared for that with our information needed for Kubernetes version 1.20. We just need to inform our cluster name inside this file for that.



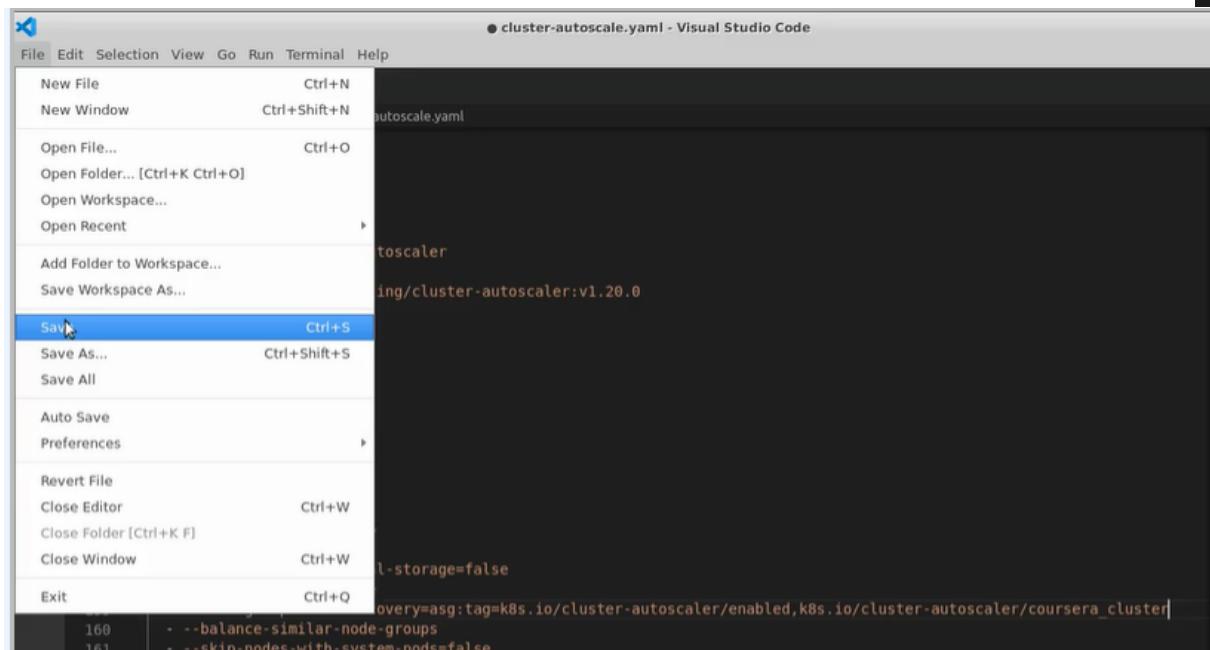
You scroll down and change this placeholder here to include our cluster name:

```
! cluster-autoscale.yaml ×  
home > rhyme > Desktop > code > ! cluster-autoscale.yaml  
135   els:  
136     pp: cluster-autoscaler  
137     otations:  
138       rometheus.io/scrape: 'true'  
139       rometheus.io/port: '8085'  
140  
141     viceAccountName: cluster-autoscaler  
142     tainers:  
143       image: k8s.gcr.io/autoscaling/cluster-autoscaler:v1.20.0  
144       name: cluster-autoscaler  
145     resources:  
146       limits:  
147         cpu: 100m  
148         memory: 300Mi  
149       requests:  
150         cpu: 100m  
151         memory: 300Mi  
152     command:  
153       - ./cluster-autoscaler  
154       - --v=4  
155       - --stderrthreshold=info  
156       - --cloud-provider=aws  
157       - --skip-nodes-with-local-storage=false  
158       - --expander=least-waste  
159       - --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/<Your Cluster Here>  
160       - --balance-similar-node-groups  
161       - --skip-nodes-with-system-pods=false  
162     volumeMounts:  
163       - name: ssl-certs  
164         mountPath: /etc/ssl/certs/ca-certificates.crt #/etc/ssl/certs/ca-bundle.crt for Amazon Linux Worker Nodes  
165         readOnly: true  
166     imes:  
167       name: ssl-certs  
168       hostPath:
```

In our case is **Coursera_cluster**:

```
149   requests:  
150     cpu: 100m  
151     memory: 300Mi  
152   command:  
153     - ./cluster-autoscaler  
154     - --v=4  
155     - --stderrthreshold=info  
156     - --cloud-provider=aws  
157     - --skip-nodes-with-local-storage=false  
158     - --expander=least-waste  
159     - --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/coursera_cluster| I  
160     - --balance-similar-node-groups  
161     - --skip-nodes-with-system-pods=false  
162   volumeMounts:  
163     - name: ssl-certs  
164       mountPath: /etc/ssl/certs/ca-certificates.crt #/etc/ssl/certs/ca-bundle.crt for Amazon Linux Worker Nodes
```

Later, we save the file:



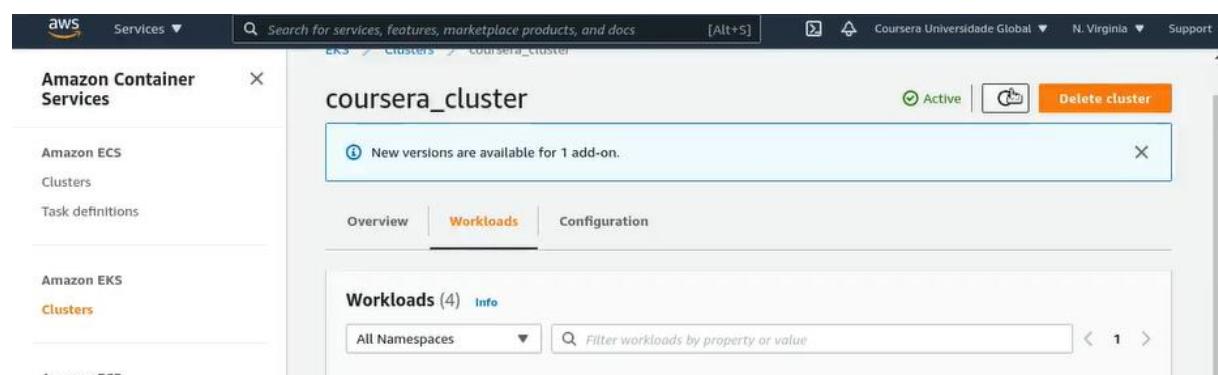
The screenshot shows the Visual Studio Code interface with the title bar "cluster-autoscale.yaml - Visual Studio Code". The "File" menu is open, and the "Save" option is highlighted with a blue selection bar. Other options in the menu include "New File", "New Window", "Open File...", "Open Folder...", "Open Workspace...", "Open Recent", "Add Folder to Workspace...", "Save Workspace As...", "Save", "Save As...", "Save All", "Auto Save", "Preferences", "Revert File", "Close Editor", "Close Folder [Ctrl+K F]", "Close Window", and "Exit". The status bar at the bottom shows the file path "toscaler/cluster-autoscaler:v1.20.0" and some command-line arguments.

Now, let's go to the terminal:

```
rhyme@ip-172-31-138-134:~/Desktop/code$ kubectl apply -f cluster-autoscale.yaml
serviceaccount/cluster-autoscaler created
clusterrole.rbac.authorization.k8s.io/cluster-autoscaler created
role.rbac.authorization.k8s.io/cluster-autoscaler created
clusterrolebinding.rbac.authorization.k8s.io/cluster-autoscaler created
rolebinding.rbac.authorization.k8s.io/cluster-autoscaler created
deployment.apps/cluster-autoscaler created
```

Our autoscaler will be deployed.

Now, let's take a look in closer workloads, on refresh it in the first:



The screenshot shows the AWS EKS Cluster Management console. On the left, there is a sidebar with "Amazon Container Services" and "Amazon EKS" sections, both with "Clusters" listed. The main area shows a cluster named "coursera_cluster" which is "Active". Below the cluster name, a message says "New versions are available for 1 add-on." There are three tabs: "Overview", "Workloads" (which is selected), and "Configuration". Under the "Workloads" tab, it says "Workloads (4)". There is a search bar with "All Namespaces" and a filter bar with "Filter workloads by property or value".



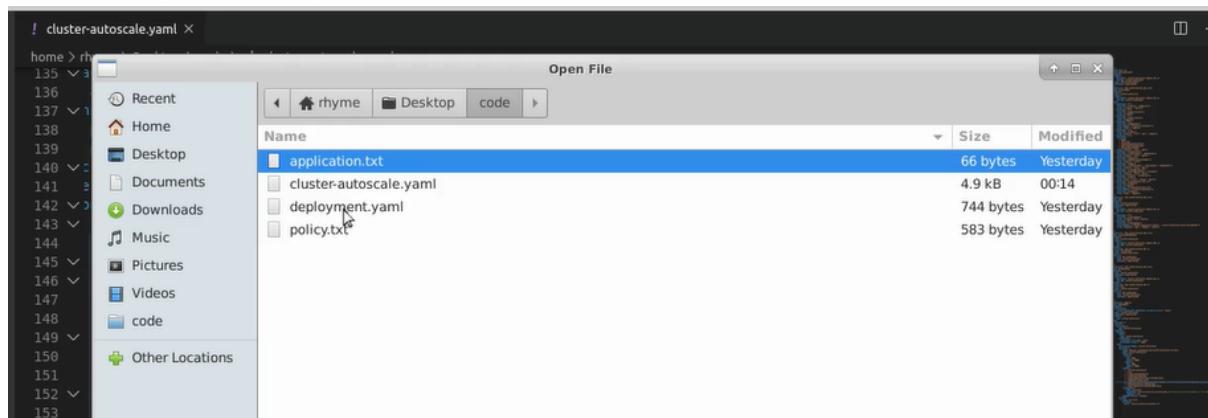
Workloads (5) [Info](#)

All Namespaces ▾ < 1 >

Name	Namespace	Type	Created	Last transition time	Pod count	Status
website	default	Deployment	16 minutes ago	16 minutes ago	5	5 Ready 0 Failed 5 Desi
coredns	kube-system	Deployment	43 minutes ago	32 minutes ago	2	2 Ready 0 Failed 2 Desi
cluster-autoscaler	kube-system	Deployment	a minute ago	a minute ago	1	1 Ready 0 Failed 1 Desi
						43

We see we have a cluster-autoscaler deployment.

To validate if autoscaler is working, we can change the replicas in our application deployment and deploy again. Back to the visual studio and open the deployment yaml file inside our code folder:



We change their replicas for 30 points:



```
! cluster-autoscale.yaml      ! deployment.yaml ●  
home > rhyme > Desktop > code > ! deployment.yaml  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: website  
5    labels:  
6      app: website  
7  spec:  
8    replicas: 30]  
9    selector:  
10   matchLabels:  
11     app: website  
12   template:  
13     metadata:  
14       labels:  
15         app: website  
16     spec:  
17       containers:  
18         name: website
```

And we save the file.

```
rhymer@ip-172-31-138-134:~/Desktop/code$ kubectl apply -f deployment.yaml  
deployment.apps/website configured  
service/website unchanged
```

The screenshot shows the Kubernetes UI interface. At the top, there are three tabs: Overview, Workloads (which is currently selected and highlighted in orange), and Configuration. Below the tabs, there's a search bar with the placeholder "Filter workloads by property or value". The main area displays a table titled "Workloads (5)". The table has columns for Name, Namespace, Type, Created, Last transition time, Pod count, and Status. One row is visible in the table:

Name	Namespace	Type	Created	Last transition time	Pod count	Status
website	default	Deployment	16 minutes ago	a few seconds ago	6	6 Ready 0 Failed 30 Desired



We have more pods being applied.

On clicking on ‘Overview’ page:

Amazon Container Services

EKS > Clusters > coursera_cluster

coursera_cluster

New versions are available for 1 add-on.

Overview Workloads Configuration

Nodes (2)

Node name	Instance type	Node Group	Created	Status
ip-172-31-13-155.ec2.internal	t3.medium	coursera_nodegroup	39 minutes ago	Ready
ip-172-31-38-179.ec2.internal	t3.medium	coursera_nodegroup	39 minutes ago	Ready

The Kubernetes cluster, we'll ask our node groups to create more virtual machines instance to deploy our pods. It can take several minutes if needed.

coursera_cluster

New versions are available for 1 add-on.

Overview Workloads Configuration

Nodes (4)

Node name	Instance type	Node Group	Created	Status
ip-172-31-13-155.ec2.internal	t3.medium	coursera_nodegroup	39 minutes ago	Ready
ip-172-31-38-179.ec2.internal	t3.medium	coursera_nodegroup	39 minutes ago	Ready
ip-172-31-38-178.ec2.internal	t3.medium	coursera_nodegroup	a few seconds ago	Ready
ip-172-31-0-146.ec2.internal	t3.medium	coursera_nodegroup	a few seconds ago	Ready

We can see now we have far Nodes, which is the maximum nodes. We define it for our hottest killer in those group configuration and in the workloads we can see we have more pods deploy it:



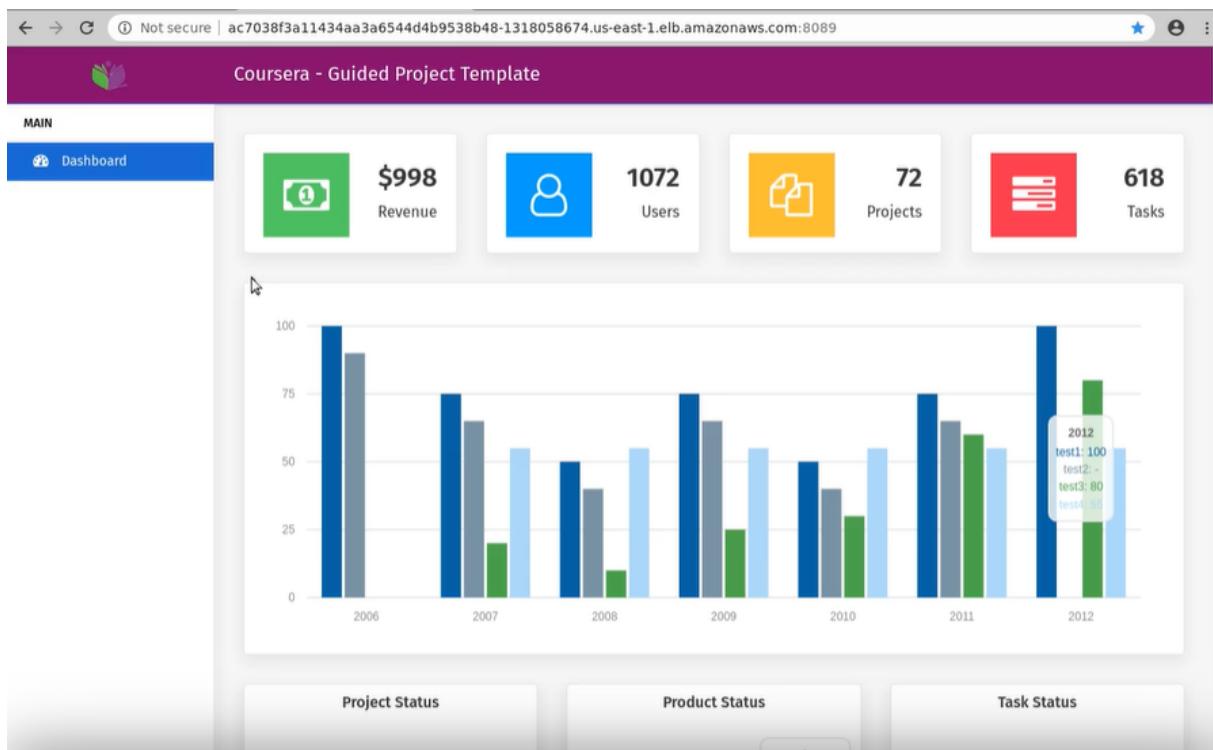
Overview Workloads Configuration

Workloads (5) Info

All Namespaces ▼ Filter workloads by property or value < 1 >

Name	Namespace	Type	Created	Last transition time	Pod count	Status
website	default	Deployment	24 minutes ago	3 minutes ago	9	9 Ready 0 Failed 30 De

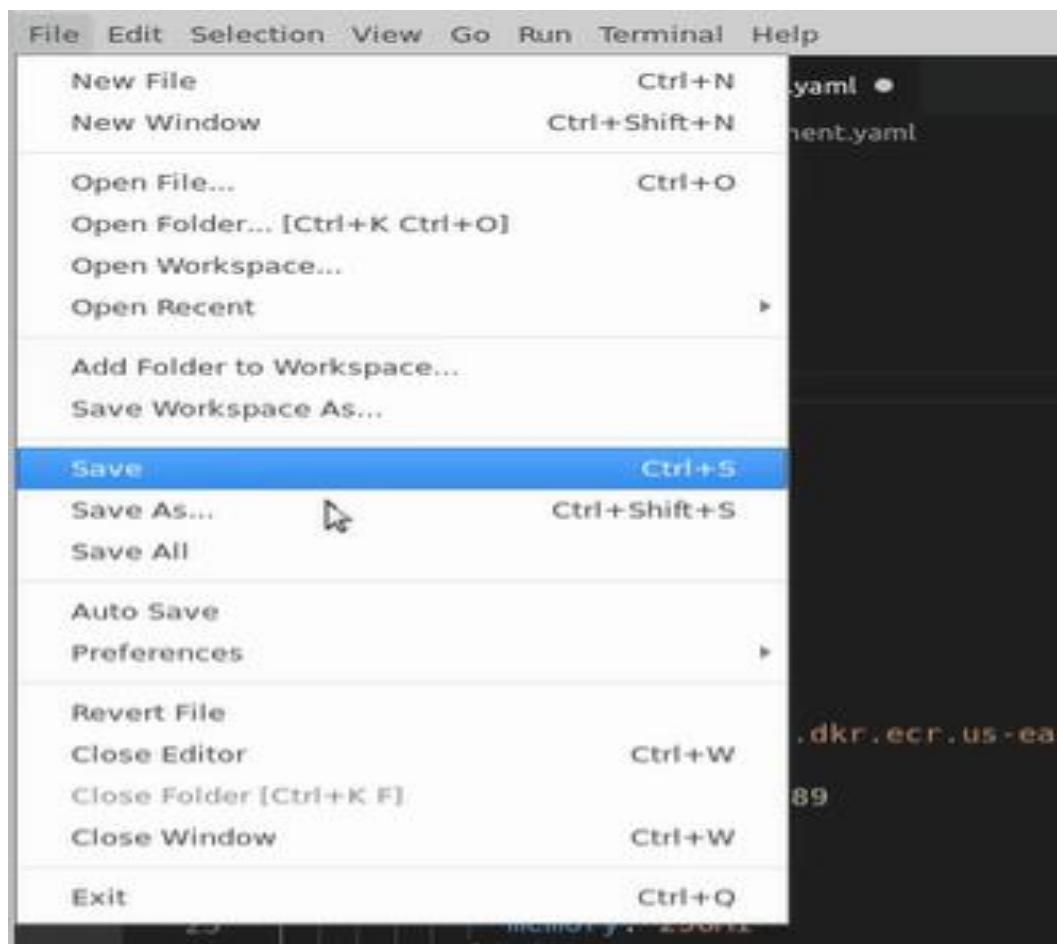
We can open a web browser tab to see if our application is still running:



What we can do now is reduce again the number of replicas for our application and see out scalar reduced the number of notes deploy it in the deployment. We change the replicas number back to five:



```
! cluster-autoscale.yaml      ! deployment.yaml •  
home > rhyme > Desktop > code > ! deployment.yaml  
1   apiVersion: apps/v1  
2   kind: Deployment  
3   metadata:  
4     name: website  
5     labels:  
6       app: website  
7   spec:  
8     replicas: 5|    I  
9     selector:  
10    matchLabels:  
11      app: website  
12    template:  
13      metadata:  
14        labels:
```



and deploy using **kubectl** too:

```
rhyme@ip-172-31-138-134:~/Desktop/code$ kubectl apply -f deployment.yaml
deployment.apps/website configured
service/website unchanged
```

And now if we refresh the cluster here:



EKS > Clusters > coursera_cluster

coursera_cluster

Active | [Edit](#) | [Delete cluster](#)

New versions are available for 1 add-on.

[Overview](#) | [Workloads](#) | [Configuration](#)

Nodes (2) [Info](#)

Node name	Instance type	Node Group	Created	Status
ip-172-31-13-155.ec2.internal	t3.medium	coursera_nodegroup	an hour ago	Ready
ip-172-31-38-179.ec2.internal	t3.medium	coursera_nodegroup	an hour ago	Ready

We have again only two nodes deployed in our cluster. Our outer scalar is working as expected.

Task 9: Clean Up the environment

In this task, we will see how to cleanup our environment to avoid further costs.

First let's take a look at some of his services.

aws Services ▾ [Search results for 'ec2'](#) Coursera University

Services (6)

Features (34)

Documentation (225,545)

Marketplace (1,233)

EC2 Virtual Servers in the Cloud

See all 6 results ▾

Top features

Dashboard Launch templates Instances Spot Instance requests Savings plans

EC2 Image Builder A managed service to automate build, customize and deploy OS Images

Some items were created by cluster, others by note group and others because we made a deployment in the service the creation.

Let's delete our deployment in our services.



```
Terminal - rhyme@ip-172-31-138-134: /  
File Edit View Terminal Tabs Help  
rhyme@ip-172-31-138-134:/ $ kubectl delete deployment website  
deployment.apps "website" deleted  
rhyme@ip-172-31-138-134:/ $ kubectl delete service website  
service "website" deleted  
rhyme@ip-172-31-138-134:/ $
```

The screenshot shows the AWS EC2 console interface. On the left, a sidebar lists navigation options like New EC2 Experience, EC2 Dashboard, Instances, and Images. The main area displays a summary of resources: 2 running instances, 0 dedicated hosts, 0 elastic IPs, 4 instances, 1 key pair, 0 load balancers, 0 placement groups, 3 security groups, 0 snapshots, and 2 volumes. A callout box provides information about creating Microsoft SQL Server Always On availability groups. On the right, the Account attributes section includes links for Supported platforms (VPC), Default VPC (vpc-5752ab2a), Settings, EBS encryption, Zones, EC2 Serial Console, Default credit specification, and Console experiments.

EKS -> Configuration -> Compute

The screenshot shows the AWS EKS console under the Amazon Container Services section. The sidebar includes options for Amazon ECS, Clusters, Task definitions, and Amazon ECR. The main view is titled 'Cluster configuration' and shows the Kubernetes version (1.20) and Platform version (eks.1). Below this, the 'Compute' tab of the Node Groups (1) section is selected, displaying a table with one row for 'coursera_nodegroup' with a desired size of 2, AMI release version 1.20.4-20210526, and status Active.



II. QCM

1. Which AWS Service can be used to store container images ?

- Elastic Kubernetes Service
- Elastic Docker Registry
- Elastic Container Registry

Correct

Correct. Elastic Container Registry is the service to store our images repositories.

2. Which Docker command do we use to build our container images?

- docker ps
- docker build
- docker run

Correct

Correct. This is the right command

3. Can we choose the size of the nodes (virtual machines) we will have in our Node Group?

- Yes
- No

Correct

Correct. You can define the virtual machine size based on EC2 predefined types

4. Which Kubernetes tool do we use to deploy containers in our Elastic Kubernetes Cluster?

- kubectl
- docker deploy

Correct

Correct. This tool could be used to deploy and manage the Kubernetes cluster



5. Which part of Kubernetes is responsible for exposing our application for users?

- Services
- Watch
- Deployments

Correct

Correct. Services define how the application will be exposed and available for users.

6. What other AWS resources does AWS create when an Elastic Kubernetes cluster is deployed?

- load balancers, public IPs, and others.
- No resources are created by AWS in an Elastic Kubernetes Service Deployment
- WebApps, Lambda, WAF and others.

Correct

Correct. AWS on behalf of the user creates a lot of other resources to support the Kubernetes architecture

7. When we want to increase or decrease the number of containers running in our Cluster, which feature we need to change in deployment.yaml file?

- Change the value in the "pod" feature.
- Change the value in the "image" feature.
- Change the value in the "replica" feature.

Correct

Correct. Changing the value of the replica feature will create a new ReplicaSet and increase or decrease the number of containers running in the cluster.

8. Which Kubernetes resource do we need to install to enable the node group scale?

- Auto Scaler
- Auto Deployment.
- Kubernetes Dashboard

Correct

Correct. You can install Kubernetes Auto Scaler to scale your node group automatically



9. After creating a Load Balancer service in our Kubernetes Clusters, which information do we use to access our application?

- DNS Name
- IP Number

Correct

Correct. Load Balancer creates a Public DNS name we can use to access our application.

10. Node groups with Auto Scaler enabled can take several minutes to scale in or scale out, is that correct?

- No.
- Yes

Correct

Correct. Any scale operation can take several minutes.