

DevOps

# GitOps: Deploying Tetris on EKS Using ArgoCD



mrcloudbook.com · 8 January 2024

Mr Cloud Book

Home Blog DevSecOps Contact About Me Testimonials

Search Blogs



Inspired From [Cloud Champ's Tutorial](#)

Github: <https://github.com/N4si/tetris-game.git>

Welcome to an exhilarating journey where we explore the cutting-edge world of GitOps, all while preparing to indulge in one of the most iconic and addictive games of all time: Tetris! In this blog, we'll unravel the mysteries of GitOps, witness its

transformational power, and set the stage for a thrilling gaming experience—all within a Kubernetes cluster deployed on Amazon's Elastic Kubernetes Service (EKS).

Buckle up, as we embark on a fascinating ride that will not only introduce you to GitOps but also allow you to enjoy the classic Tetris game on a Kubernetes platform. Let's get started and see how DevOps and gaming collide in this exciting adventure!

## Contents [\[ hide \]](#)

- [STEP 1: Create IAM Roles](#)
- [Step 2: Create EKS Cluster](#)
- [Step 3: ARGOCD SETUP](#)
- [Login](#)
- [Step 4: Change the version of the Game](#)
- [Step 5: Termination](#)

## STEP 1: Create IAM Roles

Let's First start creating two IAM roles one is for Cluster and another is for Nodegroup

Go to Aws console and search for IAM

You will be redirected to the IAM dashboard

Click "Roles"

The screenshot shows the AWS IAM Dashboard. On the left, there's a sidebar with a search bar and sections for Dashboard, Access management (with Roles highlighted), and Access reports. The main area has two panels: 'Security recommendations' (with one item) and 'IAM resources' (listing resources in the account). A red box highlights the 'Roles' section in the sidebar.

Click “Create role”

The screenshot shows the 'Roles' page in the IAM service. The sidebar highlights the 'Roles' section. The main area shows a table of existing roles and a prominent 'Create role' button, which is highlighted with a red box.

Click “Allow AWS services like EC2, Lambda, or others to perform actions in this account.”

The screenshot shows the 'Select trusted entity' step in the role creation wizard. It displays four options: 'AWS service' (selected and highlighted with a red box), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below these, a 'Use case' section is shown with the text: "Allow an AWS service like EC2, Lambda, or others to perform actions in this account."

## Click “Choose a service or use case”

<input checked="" type="radio"/> AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.	<input type="radio"/> AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.	<input type="radio"/> Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
<input type="radio"/> SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.	<input type="radio"/> Custom trust policy Create a custom trust policy to enable others to perform actions in this account.	

### Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
<i>Choose a service or use case</i>

Type “EKS”

Click this radio button with EKS-Cluster

Use case Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
Service or use case <input type="text" value="EKS"/>
Choose a use case for the specified service. Use case <input checked="" type="radio"/> EKS Allows EKS to manage clusters on your behalf. <input type="radio"/> EKS - Cluster Allows access to other AWS service resources that are required to operate clusters managed by EKS. <input type="radio"/> EKS - Nodegroup Allows EKS to manage nodegroups on your behalf. <input type="radio"/> EKS - Fargate pod Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate. <input type="radio"/> EKS - Fargate profile Allows EKS to run Fargate tasks.

Click “Next” and you will directly redirect to policy and click Next ( we have only one policy for it and it selects by default for EKS) that is `AmazonEKSClusterPolicy`

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

**Add permissions**

**Permissions policies (1) Info**  
The type of role that you selected requires the following policy.

Policy name	Type
AmazonEKSClusterPolicy	AWS managed

▶ Set permissions boundary - *optional*

Cancel Previous Next

Click the “Role name” field and provide the name ( `myAmazonEKSClusterRole` )

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

**Name, review, and create**

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
**myAmazonEKSClusterRole**

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

**Description**  
Add a short explanation for this role.  
Allows access to other AWS service resources that are required to operate clusters managed by EKS.

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

Cancel Previous Next

Click “Create role”

**Step 2: Add permissions**

**Permissions policy summary**

Policy name	Type	Attached as
AmazonEKSClusterPolicy	AWS managed	Permissions policy

**Step 3: Add tags**

**Add tags - optional Info**  
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 new tag**  
You can add up to 50 more tags.

Cancel Previous **Create role**

A cluster role is created.

## Now Create a Role for NodeGroup

Click “Create role”

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and a search bar. The main area shows a table of existing roles with columns for 'Role name', 'Trusted entities', and 'Last activity'. A red box highlights the 'Create role' button at the top right of the table.

Role name	Trusted entities	Last activity
<a href="#">AWSServiceRoleForAmazonEKS</a>	AWS Service: eks (Service-Linked Role)	
<a href="#">AWSServiceRoleForSupport</a>	AWS Service: support (Service-Linked Role)	
<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS Service: trustedadvisor (Service-Linked Role)	
<a href="#">myAmazonEKSClusterRole</a>	AWS Service: eks	

Click “Allow AWS services like EC2, Lambda, or others to perform actions in this account.”

The screenshot shows the 'Select trusted entity' step of the 'Create role' wizard. On the left, a sidebar lists 'Step 1 Select trusted entity', 'Step 2 Add permissions', and 'Step 3 Name, review, and create'. The main area is titled 'Select trusted entity' and shows a 'Trusted entity type' section with four options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. A red box highlights the 'AWS service' option.

Click “Choose a service or use case”

**Trusted entity type**

- AWS service  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy  
Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**

*Choose a service or use case*

Click “EC2”

**Select trusted entity**

Step 2  
Add permissions

Step 3  
Name, review, and create

**Commonly used services**

**EC2** (highlighted with a red box)

Lambda

**Other services**

- Amazon EMR Serverless
- Amazon OpenSearch Service
- Amazon Grafana
- Amplify
- API Gateway
- AppFabric
- Application Auto Scaling
- Application Discovery Service
- Application Migration Service
- AppStream 2.0
- AppSync

*Choose a service or use case*

**Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Click “Next”

Click the “Search” field.

The screenshot shows the AWS IAM 'Create role' wizard. It's on Step 2: 'Add permissions'. The search bar at the top of the list of policies is highlighted with a red box. The list shows several AWS managed policies:

Policy name	Type	Description
<a href="#">AdministratorAccess</a>	AWS managed - job function	-
<a href="#">AdministratorAccess-Amplify</a>	AWS managed	-
<a href="#">AdministratorAccess-AWSElasticBeanst...</a>	AWS managed	-
<a href="#">AlexaForBusinessDeviceSetup</a>	AWS managed	-
<a href="#">AlexaForBusinessFullAccess</a>	AWS managed	-
<a href="#">AlexaForBusinessGatewayExecution</a>	AWS managed	-

Search these Policy Names and make it check (I already have these in it)

AmazonEC2ContainerRegistryReadOnly

AmazonEKS\_CNI\_Policy

AmazonEBSCSIDriverPolicy

AmazonEKSWorkerNodePolicy

<input checked="" type="checkbox"/>	<input type="checkbox"/>	<a href="#">AmazonEC2ContainerRegistryReadOnly</a>	AWS managed	Provides read-only access to Amazon E...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<a href="#">AmazonEKS_CNI_Policy</a>	AWS managed	This policy provides the Amazon VPC CNI...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<a href="#">AmazonEBSCSIDriverPolicy</a>	AWS managed	IAM Policy that allows the CSI driver servi...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<a href="#">AmazonEKSWorkerNodePolicy</a>	AWS managed	This policy allows Amazon EKS worker ...

Click "Next"

Click the "Role name" field.

Add Role name as myAmazonNodeGroupPolicy

Click "Create role"

Permissions policy summary			
Policy name	Type	Attached as	
<a href="#">AmazonEBSCSI Driver Policy</a>	AWS managed	Permissions policy	
<a href="#">AmazonEC2 Container Registry Read Only</a>	AWS managed	Permissions policy	
<a href="#">AmazonEKS_CNI_Policy</a>	AWS managed	Permissions policy	
<a href="#">AmazonEKSWorkerNodePolicy</a>	AWS managed	Permissions policy	

### Step 3: Add tags

#### Add tags - optional Info

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 new tag](#)

You can add up to 50 more tags.

[Cancel](#)

[Previous](#)

**Create role**

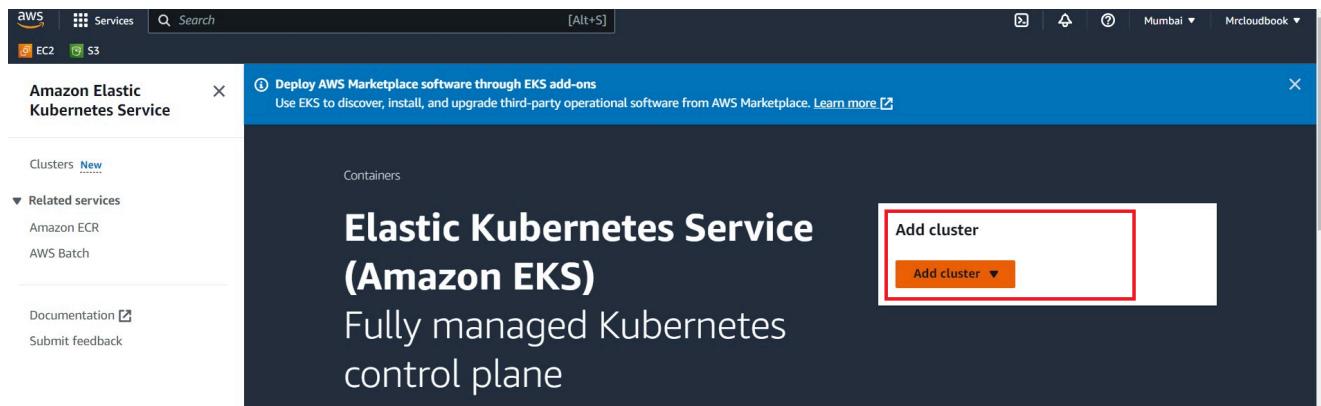
NodeGroup Role is created.

## Step 2: Create EKS Cluster

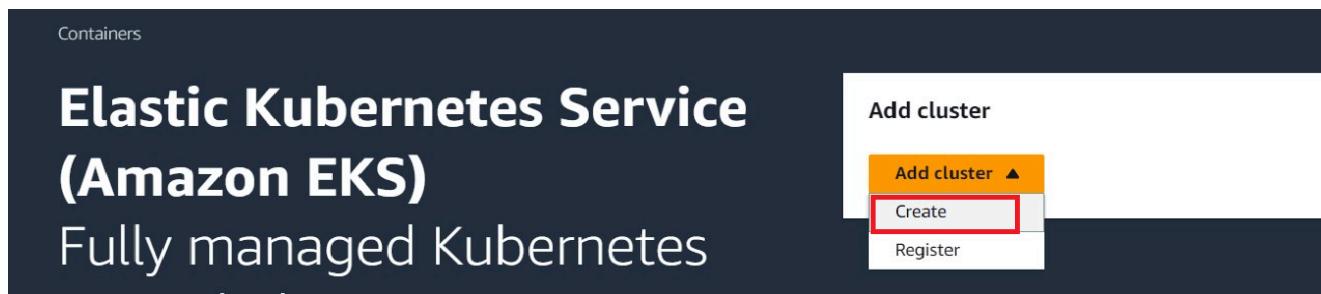
Click the “Search” field and search For EKS or select directly Elastic Kubernetes Service on the Recently visited tab

The screenshot shows the AWS Console Home page. At the top, there is a search bar with the placeholder "Search" and a keyboard shortcut "[Alt+S]". Below the search bar, there are service icons for EC2 and S3. The main area is titled "Console Home" with a "Recently visited" section. This section contains a list of services, with "Elastic Kubernetes Service" highlighted by a red box. Other services listed include EC2, Billing, S3, Support, VPC, and Elastic Container Registry. To the right of the recently visited section, there are icons for Elastic Container Service, EFS, Amazon EventBridge, and Simple Notification Service. At the bottom of the page, there is a link "View all services".

Click “Add cluster”

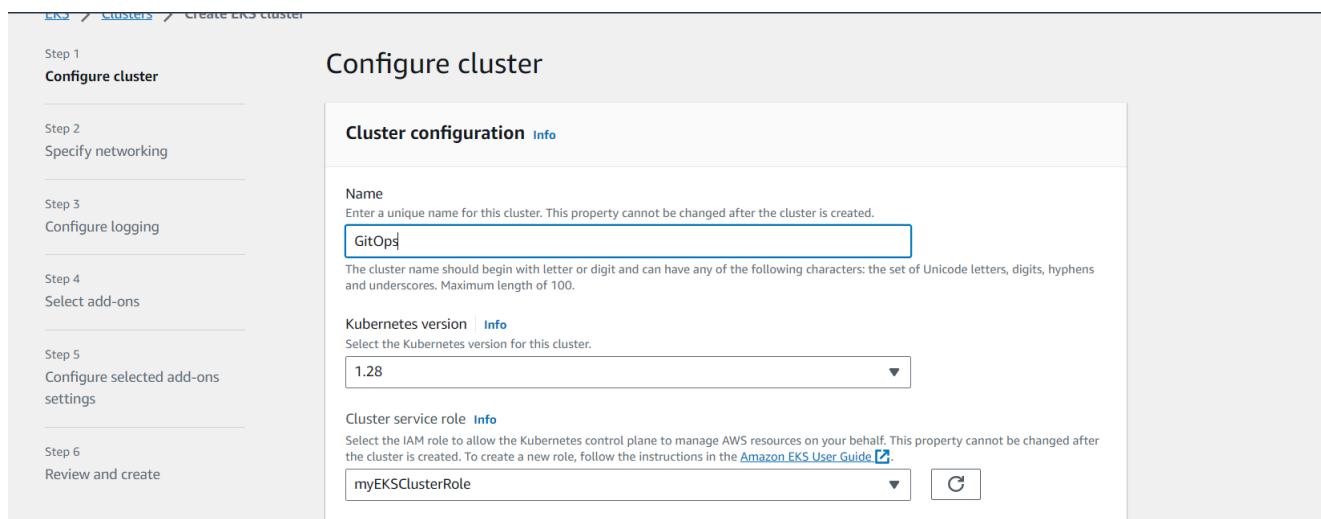


Click “Create”



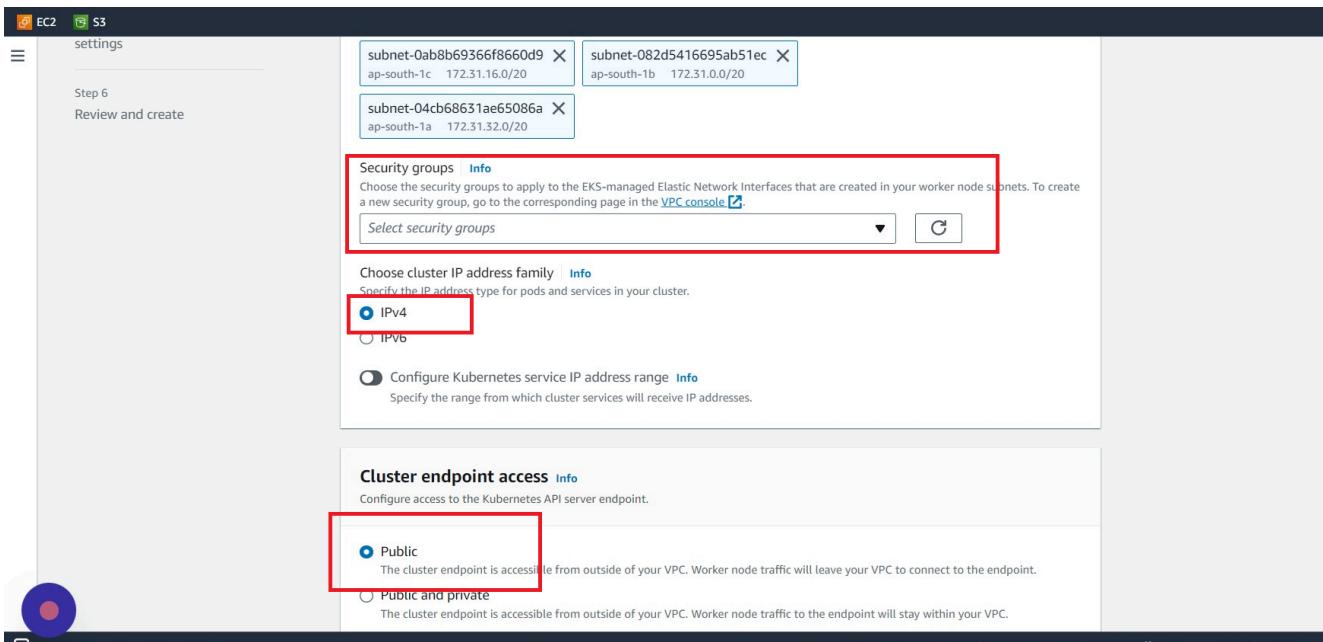
Click the “Name” field and enter a unique name for the cluster that is anything you want. For example, I used Cloud and version 1.28

Click “myAmazonEKSClusterRole” which is created in step 1.

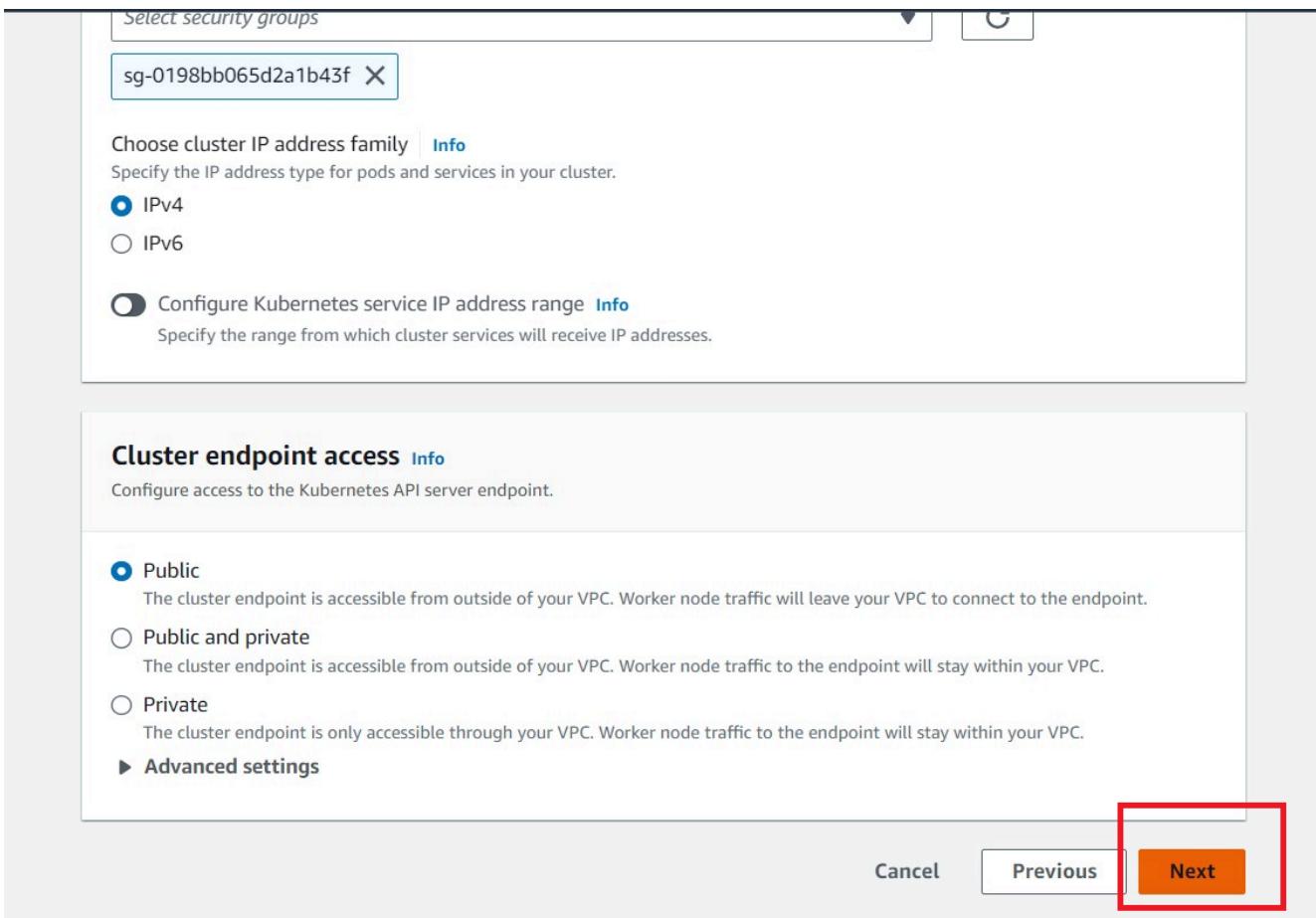


Click “Next”

Click “Select security groups” and Use the existing security group or create a new security Group



Click “Next”



Click “Next”

No changes Click “Next” (Default no need to change anything)

No changes Click “Next” (Default no need to change anything)

Click “Create”

It takes 15 minutes to create.

EKS > Clusters > GitOps

## GitOps

**Cluster info** [Info](#)

Kubernetes version <a href="#">Info</a> 1.28	Status <a href="#">Active</a>	Provider EKS
---	-------------------------------	-----------------

[Overview](#) [Resources](#) [Compute](#) [Networking](#) [Add-ons](#) [Authentication](#) [Logging](#) [Update history](#) [Tags](#)

### Details

API server endpoint <a href="https://B3C6EC80CD9BACB88F8236EE0C2D393.3.gr7.ap-south-1.eks.amazonaws.com">https://B3C6EC80CD9BACB88F8236EE0C2D393.3.gr7.ap-south-1.eks.amazonaws.com</a>	OpenID Connect provider URL <a href="https://oidc.eks.ap-south-1.amazonaws.com/id/B3C6EC80CD9BACB88F8236EE0C2D393">https://oidc.eks.ap-south-1.amazonaws.com/id/B3C6EC80CD9BACB88F8236EE0C2D393</a>	Created <a href="#">7 minutes ago</a>
Certificate authority <a href="#">View</a>	Cluster IAM role ARN <a href="arn:aws:iam::672618677785:role/myEKSClusterRole">arn:aws:iam::672618677785:role/myEKSClusterRole</a>	Cluster ARN <a href="#">arn:aws:eks:ap-south-1:672618677785:cluster/GitOps</a>
Platform version <a href="#">Info</a>		

Once your Cluster up to active status

Click “Compute”

EKS > Clusters > GitOps

## GitOps

**Cluster info** [Info](#)

Kubernetes version <a href="#">Info</a> 1.28	Status <a href="#">Active</a>	Provider EKS
---	-------------------------------	-----------------

[Overview](#) [Resources](#) [Compute](#) [Networking](#) [Add-ons](#) [Authentication](#) [Logging](#) [Update history](#) [Tags](#)

Click on “Add node group”

The screenshot shows the AWS EKS Node Groups management interface. On the left, there's a sidebar with links for EC2, S3, Amazon Elastic Kubernetes Service, Clusters (New), Related services (Amazon ECR, AWS Batch), Documentation, and Submit feedback. The main area has a search bar at the top labeled 'Filter Nodes by property or value'. Below it is a table header for 'Node name', 'Instance type', 'Node group', 'Created', and 'Status'. A message 'No Nodes' indicates the cluster does not have any nodes. In the middle section, there's another table header for 'Node groups (0)', with columns for 'Group name', 'Desired size', 'AMI release version', 'Launch template', and 'Status'. A message 'No node groups' states that the cluster does not have any node groups. At the bottom, there are 'Edit', 'Delete', and 'Add node group' buttons, with the 'Add node group' button being highlighted with a red box.

Click the “Name” field.

Write any Name you want ( NodeGroup )

Select the Role that was created for the node Group in Step 1

The screenshot shows the 'Node group configuration' step of a wizard. On the left, a sidebar lists 'Step 2: Set compute and scaling configuration', 'Step 3: Specify networking', and 'Step 4: Review and create'. The main area has a heading 'Node group configuration' with a note: '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 'NodeGroup', which is highlighted with a red box. A note below says: 'Assign a unique name for this node group. The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.' To the right is a 'Node IAM role' dropdown containing 'myEksWorkernodeRole', also highlighted with a red box. A note below says: '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.' A 'Learn more' link is provided. At the bottom, there's a 'Launch template' section with a note: 'These properties cannot be changed after the node group is created.'

Click “Next”

On the next page remove t3.medium and add t2.medium as instance type.

**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: 2 vCPUs Memory: 4 GiB Network: Up to 5 Gigabit Max ENI: 3 Max IPs: 18

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

**Node group scaling configuration**

Select t2.medium

**Configure node group**

Step 2  
**Set compute and scaling configuration**

Step 3  
Specify networking

Step 4  
Review and create

**Node group compute configuration**

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

**t2.medium** X  
vCPU: 2 vCPUs Memory: 4 GiB Network: Low to Moderate Max ENI: 3 Max IPs: 18

**t2.micro**  
vCPU: 1 vCPU Memory: 1 GiB Network: Low to Moderate Max ENI: 2 Max IPs: 4

**t2.nano**  
vCPU: 1 vCPU Memory: 0.5 GiB Network: Low to Moderate Max ENI: 2 Max IPs: 4

**t2.small**  
vCPU: 1 vCPU Memory: 2 GiB Network: Low to Moderate

Select

Click "Next"

2 nodes

**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.  
2 nodes

**Node group update configuration** Info

**Maximum unavailable**  
Set the maximum number or percentage of unavailable nodes to be tolerated during the node group version update.

Number  
Enter a number

Percentage  
Specify a percentage

**Value**  
1 node

Cancel Previous **Next**

Click “Next”

EKS > Clusters > Cloud > Add node group

Step 1 [Configure node group](#)

Step 2 [Set compute and scaling configuration](#)

Step 3 **Specify networking**

Step 4 [Review and create](#)

**Specify networking**

**Node group network configuration**  
These properties cannot be changed after the node group is created.

**Subnets** Info  
Specify the subnets in your VPC where your nodes will run. To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets ▾

subnet-0ab8b69366f8660d9 X subnet-082d5416695ab51ec X

subnet-04cb68631ae65086a X

Configure remote access to nodes Info

Cancel Previous **Next**

Click “Create”

**Step 3: Networking**

Node group network configuration

Subnets

- subnet-0ab8b69366f8660d9
- subnet-082d5416695ab51ec
- subnet-04cb68631ae65086a

Configure remote access to nodes

off

Cancel Previous Create

Node Groups will take some time to create.

EKS > Clusters > GitOps > Node group: NodeGroup

### NodeGroup

Node group configuration [Info](#)

Kubernetes version 1.28	AMI type <a href="#">Info</a> AL2_x86_64	Status <span style="color: green;">Active</span>
AMI release version <a href="#">Info</a> 1.28.1-20231002	Instance types t2.medium	Disk size 20 GiB

Details **Nodes** Health issues (0) Kubernetes labels Update config Kubernetes taints Update history Tags

**Nodes (2) [Info](#)**

Node name	Instance type	Node group	Created	Status
ip-172-31-40-129.ap-south-1.compute.internal	t2.medium	<a href="#">NodeGroup</a>	Created <span style="color: green;">a minute ago</span>	<span style="color: green;">Ready</span>
ip-172-31-5-141.ap-south-1.compute.internal	t2.medium	<a href="#">NodeGroup</a>	Created <span style="color: green;">a minute ago</span>	<span style="color: green;">Ready</span>

Worker nodes created.

## Step 3: ARGOCD SETUP

Click on the AWS cloud shell icon on the top right

Amazon Elastic Kubernetes Service

Clusters New

Related services

- Amazon ECR
- AWS Batch

Documentation

**NodeGroup**

**Node group configuration** Info

Kubernetes version 1.28	AMI type <small>Info</small> AL2_x86_64	Status <span style="color: green;">Active</span>
AMI release version <small>Info</small> 1.28.1-20231002	Instance types t2.medium	Disk size 20 GiB

**CLICK HERE FOR CLOUD SHELL**

click on connect

Better to open in a new tab

First set context by providing the following command

```
aws eks update-kubeconfig --name EKS_CLUSTER_NAME --region CLUSTER_REGION
#example
aws eks update-kubeconfig --name GitOps --region ap-south-1
```

**AWS CloudShell**

ap-south-1

```
[cloudshell-user@ip-10-2-87-125 ~]$ aws eks update-kubeconfig --name GitOps --region ap-south-1
Added new context arn:aws:eks:ap-south-1:6261867785:cluster/GitOps to /home/cloudshell-user/.kube/config
[ccloudshell-user@ip-10-2-87-125 ~]$
```

Check for Nodes

```
kubectl get nodes
```

```
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$ kubectl get nodes  
NAME                      STATUS   ROLES      AGE     VERSION  
ip-172-31-40-129.ap-south-1.compute.internal   Ready    <none>   4m8s   v1.28.1-eks-43840fb  
ip-172-31-5-141.ap-south-1.compute.internal   Ready    <none>   4m28s  v1.28.1-eks-43840fb  
[cloudshell-user@ip-10-2-87-125 ~]$
```

Check for pods, You will get no resources found.



```
kubectl get pods
```



**AWS CloudShell**

**ap-south-1**

```
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$ kubectl get pods  
No resources found in default namespace.  
[cloudshell-user@ip-10-2-87-125 ~]$
```

Let's install ArgoCD

### ARGOCD INSTALLATION LINK

You will redirected to this page

This workshop has been deprecated and archived. The new Amazon EKS Workshop is now available at [www.eksworkshop.com](http://www.eksworkshop.com).

## ArgoCD Architecture

All those components could be installed using a manifest provided by the Argo Project: use the below commands

<https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml>

```
[cloudshell-user@ip-10-2-87-125 ~]$ kubectl create namespace argocd
namespace/argocd created
[cloudshell-user@ip-10-2-87-125 ~]$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml
customresourcedefinition.apirextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apirextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apirextensions.k8s.io/applications.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-server created
configmap/argocd-cm created
configmap/argocd-cmd-params-cm created
configmap/argocd-gpg-keys-cm created
configmap/argocd-notifications-cm created
configmap/argocd-rbac-cm created
configmap/argocd-ssh-known-hosts-cm created
configmap/argocd-tls-certs-cm created
```

## COMMANDS ARGOCD

By default, argocd-server is not publicly exposed. For this project, we will use a Load Balancer to make it usable:



```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBa'
```



```
aws CloudShell
ap-south-1

[cloudshell-user@ip-10-2-87-125 ~]$ 
[cloudshell-user@ip-10-2-87-125 ~]$ 
[cloudshell-user@ip-10-2-87-125 ~]$ 
[cloudshell-user@ip-10-2-87-125 ~]$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
service/argocd-server patched
[cloudshell-user@ip-10-2-87-125 ~]$ 
```

One load balancer will created in the AWS

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
a646906ffc5264bf8807 d85414d5b69f	a646906ffc5264bf8807d8...	-	vpc-0f6bdd74ced5c07c0	3 Availability Zones	classic	October 2023 (UTC+0)

Wait about 2 minutes for the LoadBalancer creation



```
:d -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostname'`
```



when you run this command, it will export the hostname of the ArgoCD server's load balancer and store it in the `ARGOCD_SERVER` environment variable, which you can then use in other commands or scripts to interact with the ArgoCD server. This can be useful when you need to access the ArgoCD web UI or interact with the server programmatically.

```
ap-south-1
[cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ export ARGOCD_SERVER=`kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostname'` [cloudshell-user@ip-10-2-87-125 ~]$ 
```

If run this command you will get the load balancer external IP



```
echo $ARGOCD_SERVER
```



```
[cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ echo $ARGOCD_SERVER
a646906ffc5264bf8807d85414d5b69f-937542690.ap-south-1.elb.amazonaws.com
[cloudshell-user@ip-10-2-87-125 ~]$ 
```

## Login

The command you provided is used to extract the password for the initial admin user of ArgoCD, decode it from base64 encoding, and store it in an environment variable named `ARGO_PWD`.



```
export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o yaml | base64 -d | jq --raw-output '.data.password' | base64 -d`
```



```
[cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o yaml | base64 -d | jq --raw-output '.data.password' | base64 -d` [cloudshell-user@ip-10-2-87-125 ~]$ 
```

If you want to see your password provide the below command



```
echo $ARGO_PWD
```



```
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$ echo $ARGO_PWD  
ky21EiuybUiMev8k  
[cloudshell-user@ip-10-2-87-125 ~]$
```

Now copy the load balancer IP and paste it into the browser

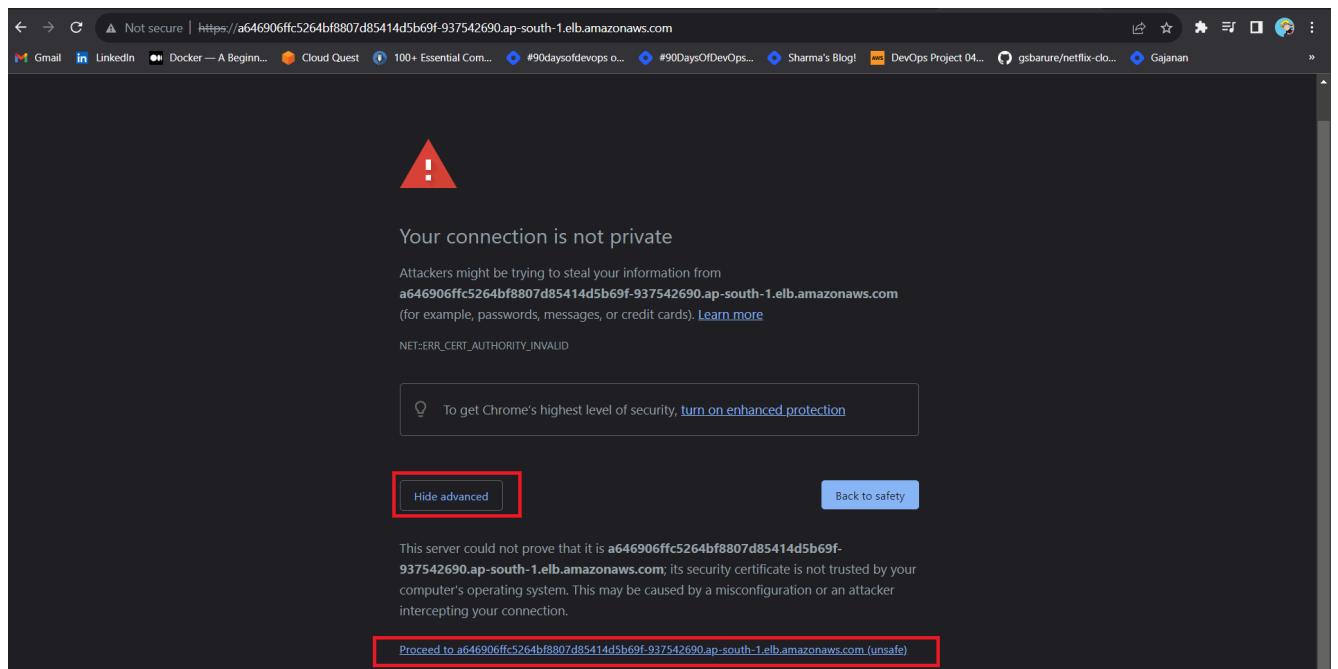


```
echo $ARGOCD_SERVER
```

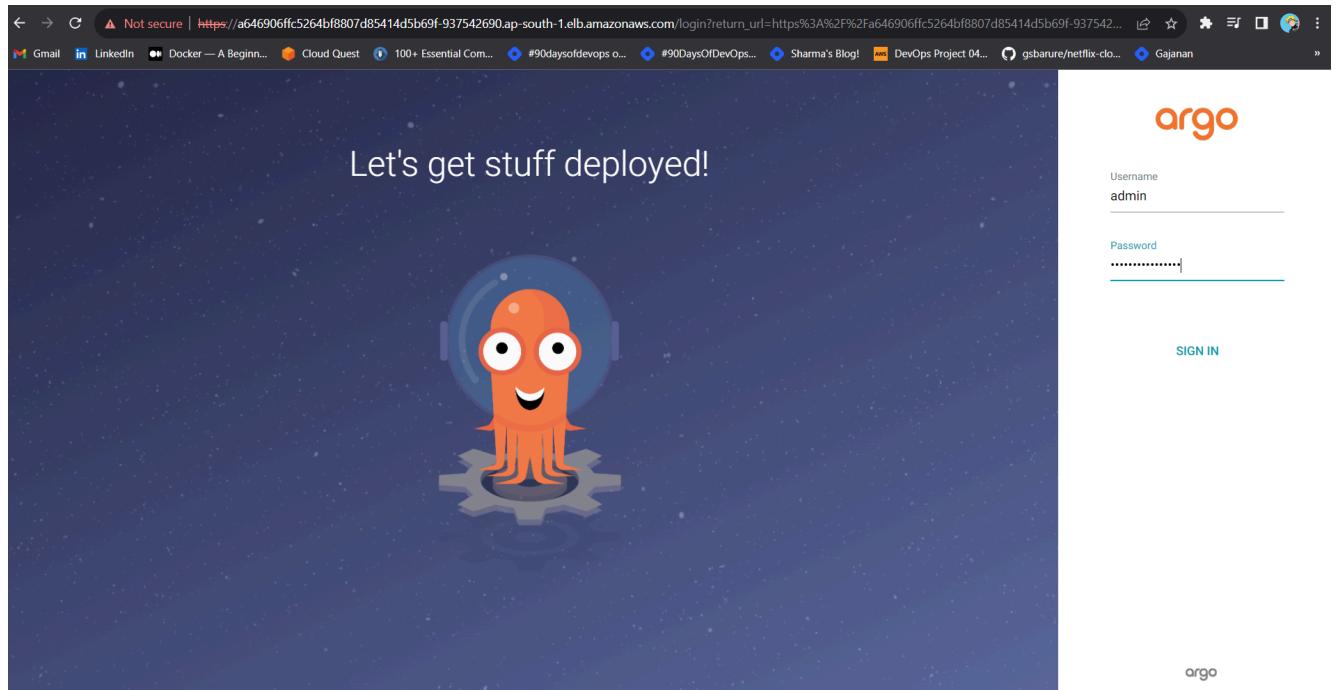


```
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$ echo $ARGOCD_SERVER  
a646906ffc5264bf8807d85414d5b69f-937542690.ap-south-1.elb.amazonaws.com  
[cloudshell-user@ip-10-2-87-125 ~]$  
[cloudshell-user@ip-10-2-87-125 ~]$
```

Now you will see this page. if you get an error click on advanced and click on proceed.



Now you will see this page and log in to ArgoCD



Username is admin

For the password, you have to provide the below command and copy it

```
[cloudshell-user@ip-10-2-87-125 ~]$ echo $ARGO_PWD
ky21EiuybUiMev8k
```

A context menu is open over the password 'ky21EiuybUiMev8k'. The menu items are:

Emoji	Win+Period
Cut	Ctrl+X
<b>Copy</b>	<b>Ctrl+C</b>
Paste	Ctrl+V
Paste as plain text	Ctrl+Shift+V
Select all	Ctrl+A

Click on Signin and you will see this page.

The screenshot shows the ArgoCD application dashboard. At the top, there's a header bar with various links like Gmail, LinkedIn, Docker — A Beginner's Guide, Cloud Quest, 100+ Essential Com..., #90daysOfDevops o..., #90DaysOfDevOps..., Sharma's Blog!, DevOps Project 04..., gsbarure/netflix-cl..., and Gajanan. Below the header is a navigation bar with icons for Applications, + NEW APP, SYNC APPS, REFRESH APPS, and a search bar. To the right of the search bar are buttons for APPLICATIONS TILES, Log out, and other navigation options. The main content area features a large circular icon with three stacked rectangles inside, representing a stack or cluster. Below the icon, the text "No applications yet" is displayed, followed by the instruction "Create new application to start managing resources in your cluster". A "CREATE APPLICATION" button is located at the bottom of this section.

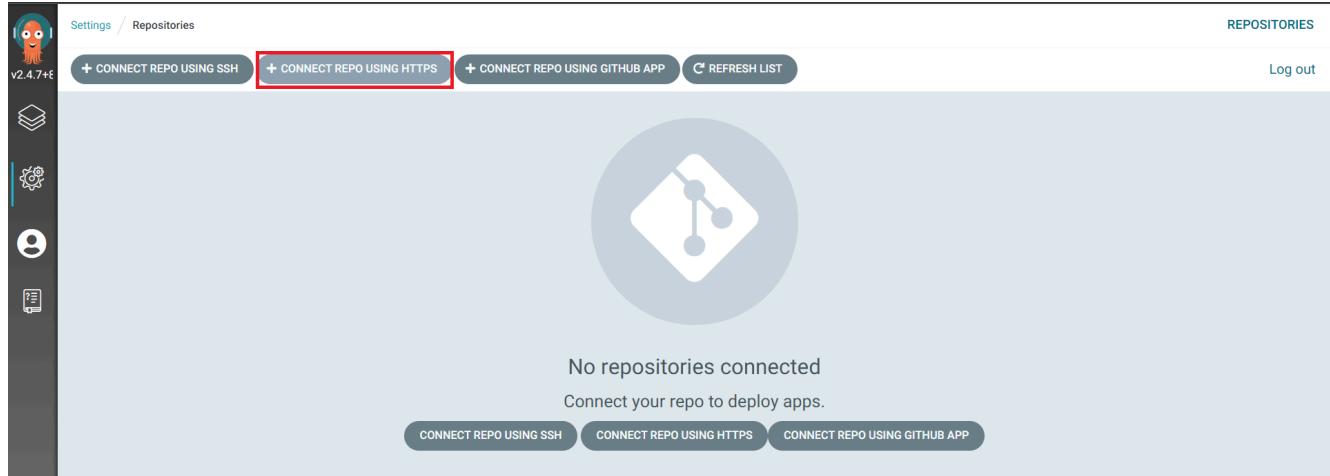
Now click on the Setting gear icon in the left side panel

This screenshot is similar to the previous one, showing the ArgoCD dashboard. However, the gear icon in the left sidebar is highlighted with a red box. The central content area is mostly empty, showing the "No applications yet" message and the "CREATE APPLICATION" button.

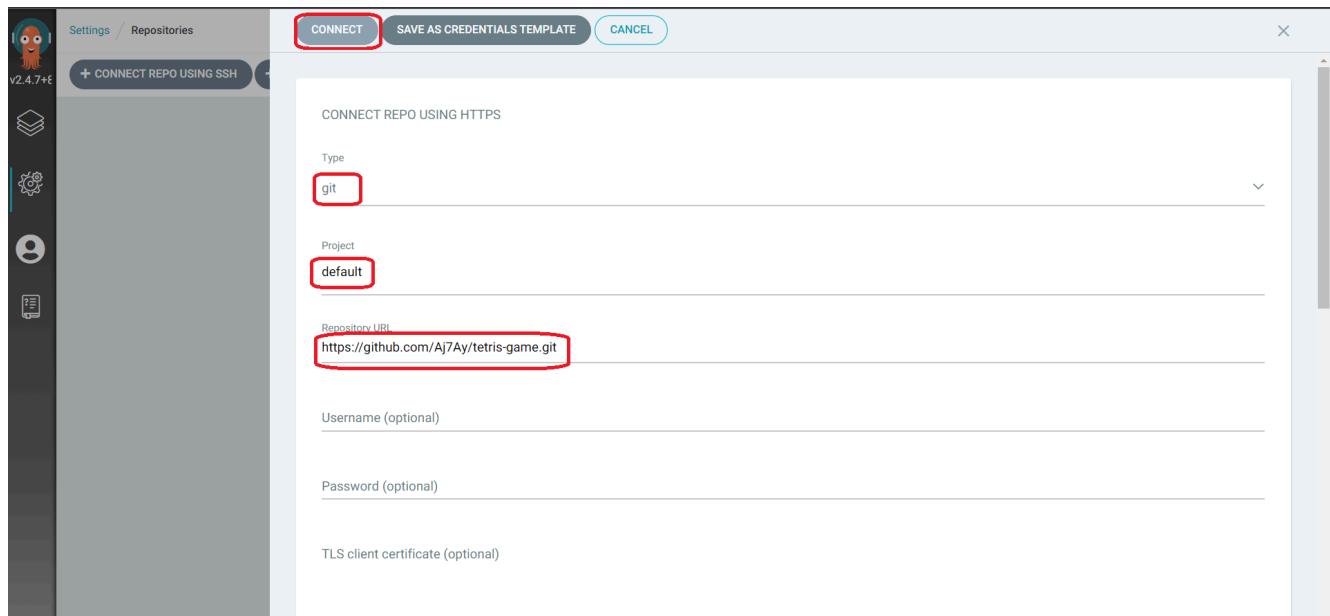
Click on Repositories

This screenshot shows the ArgoCD Settings page. The left sidebar has icons for Applications, Settings (which is selected and highlighted with a red box), Certificates, GnuPG keys, and Clusters. The main content area contains four cards: "Repositories" (Configure connected repositories), "Certificates" (Configure certificates for connecting Git repositories), "GnuPG keys" (Configure GnuPG public keys for commit verification), and "Clusters" (Configure connected Kubernetes clusters). The "Repositories" card is also highlighted with a red box.

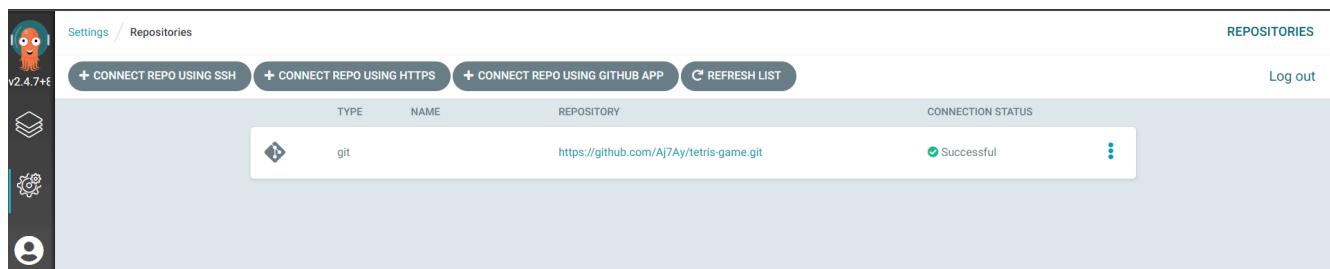
Now click on Connect Repo Using HTTPS



Add Github details, Type as git, Project as default and provide the GitHub URL of this project and click on connect



You will get Connection Status as Successful



Click on Manage Your application

The screenshot shows the ArgoCD web interface. At the top, there are buttons for connecting repos via SSH, HTTPS, or GitHub App, and a refresh list button. On the right, there's a 'REPOSITORIES' link and a 'Log out' button. The main area is titled 'REPOSITORIES' and shows a single repository entry: 'https://github.com/Aj7Ay/tetris-game.git' with a 'Successful' connection status. A red box highlights the 'Manage your applications, and diagnose health problems.' button.

You will see this page and click on New App

The screenshot shows the ArgoCD interface with the 'Applications' tab selected. On the left, there are icons for Settings, Repositories, Applications, and Logs. The main area has buttons for '+ NEW APP', 'SYNC APPS', and 'REFRESH APPS'. A search bar and a 'Log out' button are also present. A red box highlights the '+ NEW APP' button.

Now provide the following details as in the image

The screenshot shows the 'CREATE' dialog for a new application. The 'GENERAL' section includes fields for 'Application Name' (set to 'tetris') and 'Project Name' (set to 'default'). Under 'SYNC POLICY', 'Automatic' is selected. There are also checkboxes for 'PRUNE RESOURCES' and 'SELF HEAL'. A red box highlights the 'Application Name' field.

The screenshot shows the 'CREATE' dialog for a new application. The 'SOURCE' section includes 'Repository URL' (set to 'https://github.com/Aj7Ay/tetris-game.git'), 'Revision' (set to 'HEAD'), and 'Path' (set to 'manifests'). The 'DESTINATION' section includes 'Cluster URL' (set to 'https://kubernetes.default.svc') and 'Namespace' (set to 'default'). A red box highlights the 'Repository URL' field.

Finally, click on Create

In the deployment file, we used the image as

tetris-game / manifests / tetris-deploy.yaml

Aj7Ay Update tetris-deploy.yaml

**Code** **Blame** 21 lines (20 loc) · 496 Bytes

```

1  # tetris-deployment.yaml
2
3  apiVersion: apps/v1
4  kind: Deployment
5  metadata:
6    name: tetris-deployment
7  spec:
8    replicas: 1 # You can adjust the number of replicas as needed
9    selector:
10      matchLabels:
11        app: tetris
12    template:
13      metadata:
14        labels:
15          app: tetris
16      spec:
17        containers:
18          - name: tetris
19            image: nasi101/tetris # Replace with the actual image tag
20            ports:
21              - containerPort: 80 # Replace with the port your Tetris game listens on

```

Now you will see a new App has been created named tetris

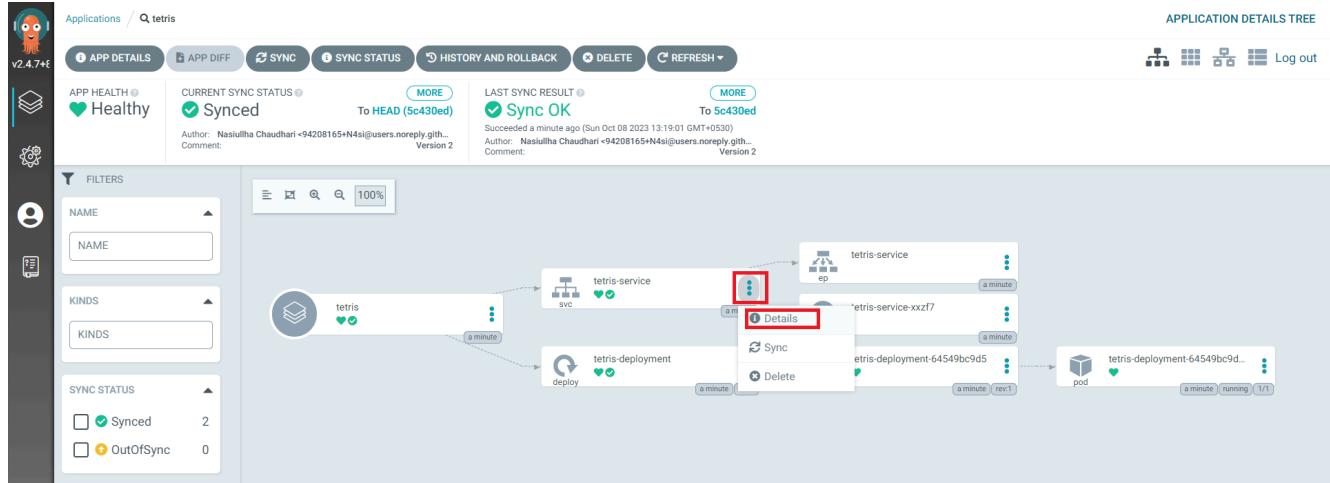
The screenshot shows the ArgoCD application dashboard. On the left, there's a sidebar with icons for Applications, New App, Sync Apps, Refresh Apps, and Log Out. Below these are sections for Filters (Favorites Only), Sync Status (Unknown: 0, Synced: 1, OutOfSync: 0), and Health Status (Unknown: 0, Progressing: 0, Suspended: 0, Healthy: 1, Degraded: 0, Missing: 0). The main area displays the 'tetris' application details in a card:

- Project: default
- Labels: (empty)
- Status: Healthy (Synced)
- Repository: https://github.com/Aj7Ay/tetris-game.git
- Target Rev.: HEAD
- Path: manifests
- Destination: in-cluster
- Namespace: default

At the bottom of the card are three buttons: SYNC, REFRESH, and DELETE.

Click on tetris

Now click on three dots beside tetris-service and click on the details



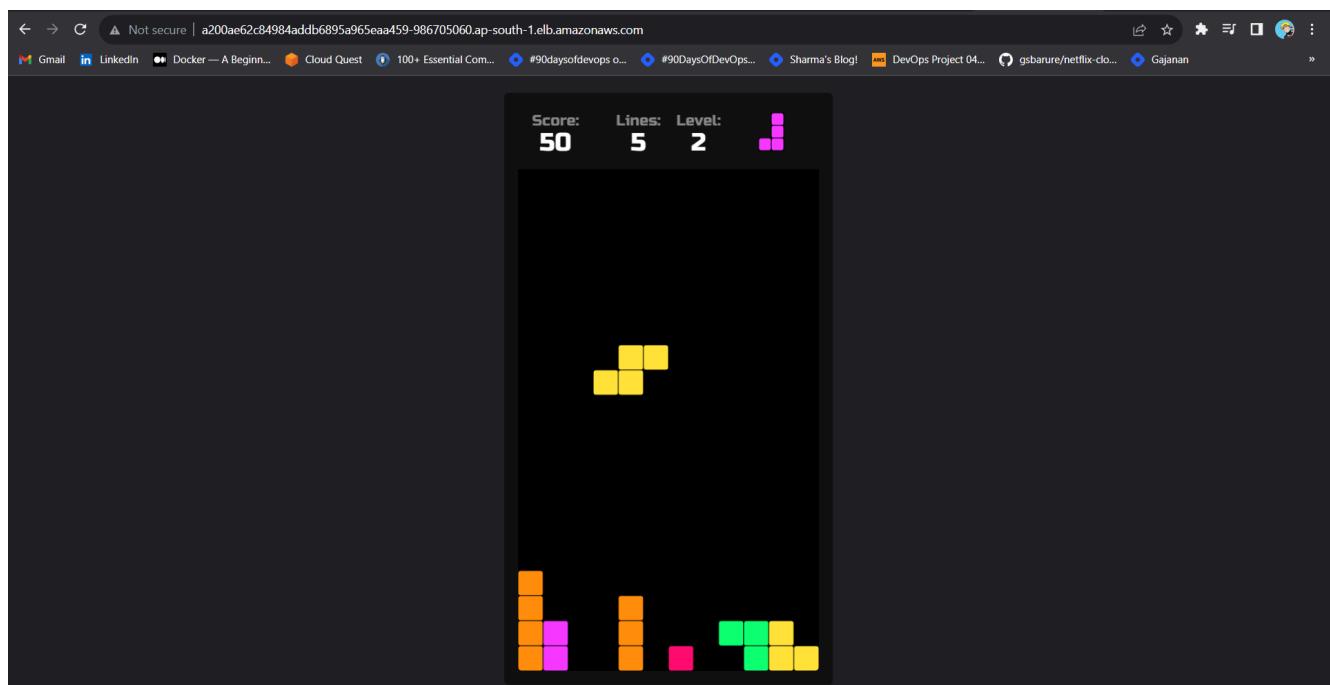
Now copy the hostname address

The screenshot shows the ArgoCD interface with the 'tetris-service' resource selected. The 'SUMMARY' tab is active, displaying the following details:

KIND	Service
NAME	tetris-service
NAMESPACE	default
CREATED_AT	10/08/2023 13:19:01
TYPE	LoadBalancer
HOSTNAMES	a200ae62c84984addb6895a965eaa459-986705060.ap-south-1.elb.amazonaws.com
STATUS	Synced
HEALTH	Healthy

A red box highlights the 'HOSTNAMES' field.

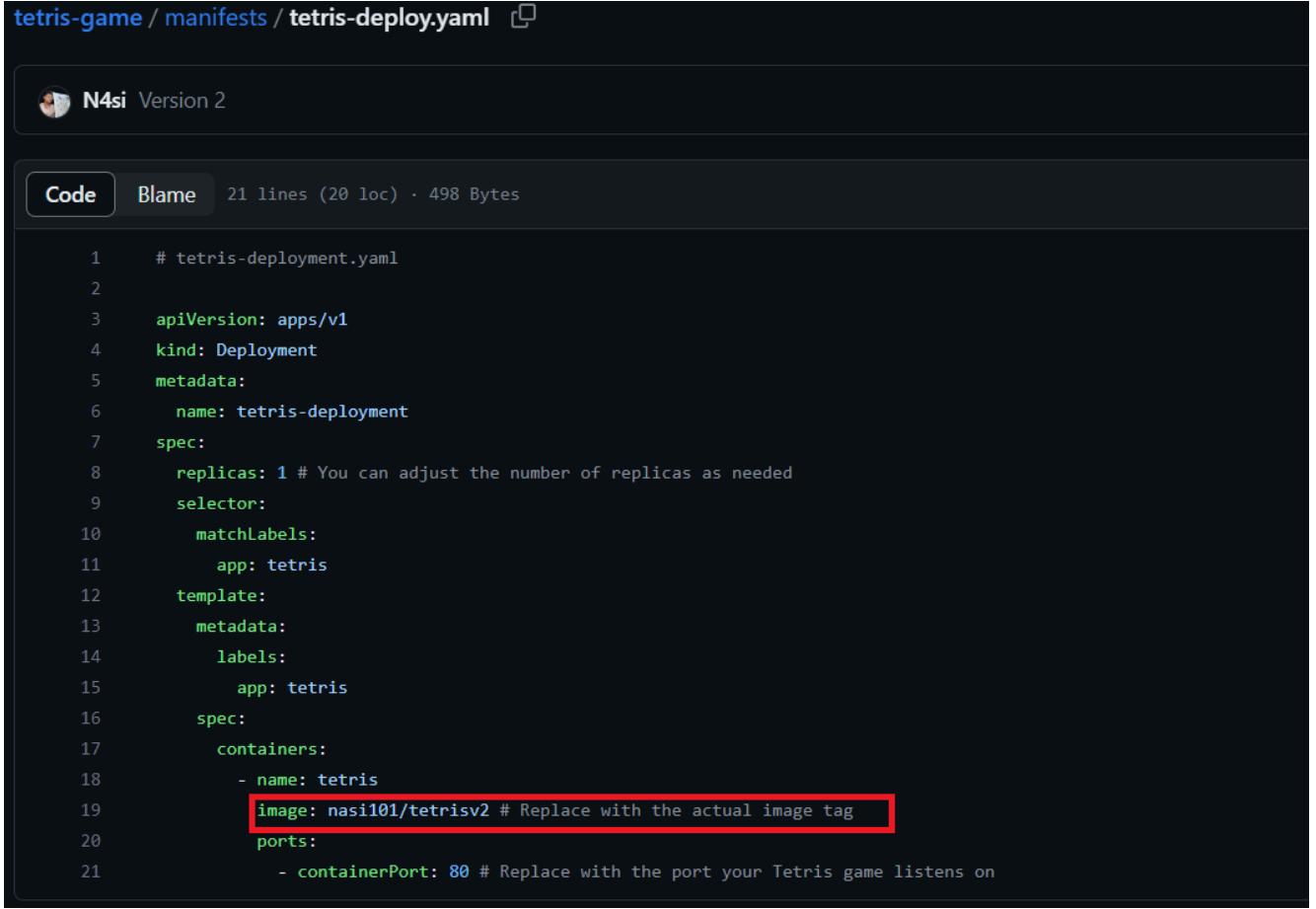
Paste it in a browser you will see this page



Now play the game of version 1.

## Step 4: Change the version of the Game

Now change the version of the game in the deployment file



```

tetris-game / manifests / tetris-deploy.yaml □

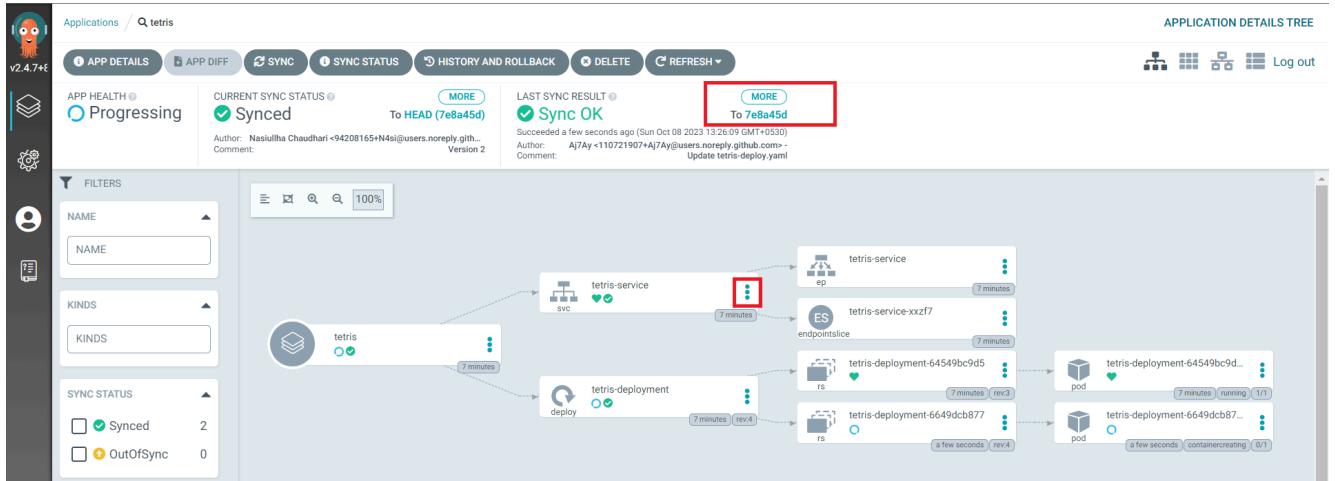
N4si Version 2

Code Blame 21 lines (20 loc) · 498 Bytes

1   # tetris-deployment.yaml
2
3   apiVersion: apps/v1
4   kind: Deployment
5   metadata:
6     name: tetris-deployment
7   spec:
8     replicas: 1 # You can adjust the number of replicas as needed
9     selector:
10       matchLabels:
11         app: tetris
12     template:
13       metadata:
14         labels:
15           app: tetris
16       spec:
17         containers:
18           - name: tetris
19             image: nasi101/tetrisv2 # Replace with the actual image tag
20             ports:
21               - containerPort: 80 # Replace with the port your Tetris game listens on

```

ArgoCD now automatically starts the build when you change anything in your Repository or the files

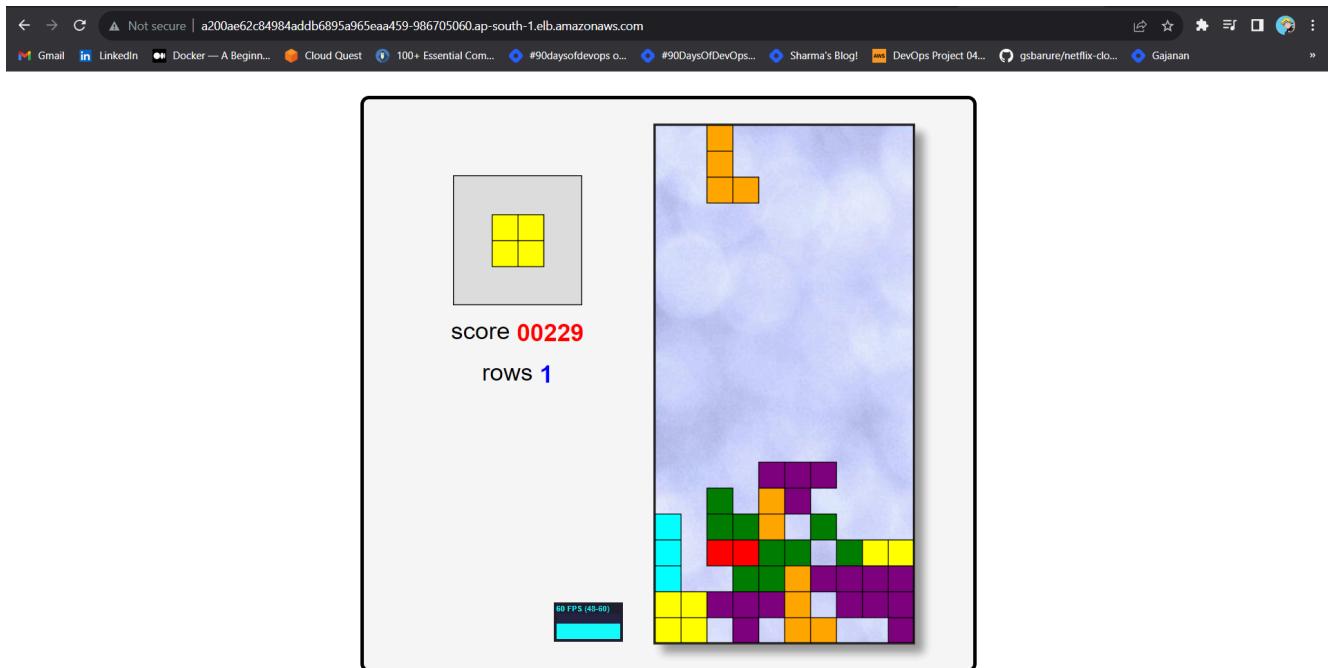


Now Last Sync id changed

Click on three dots at service and copy the hostname address

KIND	Service
NAME	tetris-service
NAMESPACE	default
CREATED_AT	10/08/2023 13:19:01
TYPE	LoadBalancer
HOSTNAMES	a200ae62c84984addb6895a965eaa459-986705060.ap-south-1.elb.amazonaws.com
STATUS	Synced
HEALTH	Healthy

Now you will see the second version of the Game



Let's Break the Highest score in Tetris.



kubectl get all



```
[cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/tetris-deployment-6649dcb877-rkxk2      1/1     Running   0          4m29s

NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes ClusterIP  10.100.0.1 <none>       443/TCP   35m
service/tetris-service LoadBalancer 10.100.69.208 a200ae62c84984addb6895a965eaa459-986705060.ap-south-1.elb.amazonaws.com 80:30627/TCP 11m

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tetris-deployment            1/1     1           1           11m

NAME           DESIRED  CURRENT   READY   AGE
replicaset.apps/tetris-deployment-64549bc9d5  0        0         0        11m
replicaset.apps/tetris-deployment-6649dcb877  1        1         1        5m4s
[cloudshell-user@ip-10-2-87-125 ~]$
```

## Step 5: Termination

Go to Cloud shell and delete service for ArgoCD load balancer



kubectl delete svc argocd-server -n argocd



```
[cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ [cloudshell-user@ip-10-2-87-125 ~]$ kubectl delete svc argocd-server -n argocd
service "argocd-server" deleted
[cloudshell-user@ip-10-2-87-125 ~]$
```

Go to AWS and manually delete the load balancer if it is not deleted.

The screenshot shows the AWS EC2 Load Balancers page. At the top, there is a breadcrumb navigation: EC2 > Load balancers. Below the header, there is a search bar labeled 'Filter Load balancers' and a set of actions buttons: 'Actions' with a dropdown arrow, 'Create load balancer', and a refresh icon. A message below the search bar says 'Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.' In the main table area, there is a single row with a checkbox, the name 'No load balancers', and a note 'You don't have any load balancers in ap-south-1'. The table has columns for Name, DNS name, State, VPC ID, Availability Zones, Type, and Data.

Delete Nodegroup First

The screenshot shows the 'NodeGroup' configuration page. At the top right, there are three buttons: 'C' (Clone), 'Edit', and 'Delete'. The 'Delete' button is highlighted with a red box. Below the buttons, the section title 'Node group configuration' has an 'Info' link. The configuration details are listed in a table:

Kubernetes version 1.28	AMI type <a href="#">Info</a> AL2_x86_64	Status <span style="color: green;">Active</span>
AMI release version <a href="#">Info</a> 1.28.1-20231002	Instance types t2.medium	Disk size 20 GiB

Below the table, there are several navigation tabs: 'Details' (highlighted with a blue underline), 'Nodes', 'Health issues (0)', 'Kubernetes labels', 'Update config', 'Kubernetes taints', 'Update history', and 'Tags'.

Provide the name that you used for your node group and click delete

The screenshot shows a modal dialog titled 'Delete: NodeGroup'. It contains two warning sections:

- Terminating this node group will stop running pods**: This section states that pods running on the node group will be terminated, and other node groups may be affected when pods are rescheduled. It also notes that if no other node groups exist, deployments will fail and services may become unavailable.
- Terminating this managed node group could affect self-managed node groups**: This section explains that deleting a managed node group that uses a specific IAM role will remove it from the aws-auth ConfigMap. It advises ensuring no self-managed node groups share the same IAM role.

At the bottom, a note says 'To confirm deletion, type the node group name in the field.' A red box highlights the input field where 'NodeGroup' is typed. The 'Delete' button at the bottom right is also highlighted with a red box.

Delete the Cluster

The screenshot shows the 'Cluster info' section of a GitOps interface. At the top right are two buttons: a blue one with a circular arrow icon and a white one labeled 'Delete cluster'. Below them is a table with three columns: 'Kubernetes version' (Info, 1.28), 'Status' (Active), and 'Provider' (EKS). Below the table is a navigation bar with tabs: Overview, Resources, Compute (selected), Networking, Add-ons, Authentication, Logging, Update history, and Tags.

Provide the name of the Cluster and delete.

A modal dialog box titled 'Delete: GitOps' is displayed. It contains a yellow warning box with an exclamation mark icon and the text 'Deleting this EKS cluster will permanently remove it'. Below it is the question 'Are you sure you want to delete this EKS cluster?'. A text input field below the warning box contains the text 'GitOps', which is highlighted with a red box. At the bottom right of the dialog are two buttons: 'Cancel' and a large orange 'Delete' button, which is also highlighted with a red box.

As we conclude our journey into the fascinating realm of GitOps and the thrilling world of Tetris on Kubernetes, let's leave you with a fun fact:

Did you know that the creator of Tetris, Alexey Pajitnov, designed the game in 1984 while working as a computer scientist at the Soviet Academy of Sciences? What started as a simple experiment turned into a worldwide gaming sensation that has captured the hearts and minds of players for decades.

Just like Tetris, GitOps continues to evolve and shape the landscape of modern DevOps, offering innovative solutions for managing complex infrastructures and applications. We hope this adventure has piqued your curiosity and inspired you to explore the limitless possibilities of GitOps further.

So, whether you're deploying code, stacking blocks in Tetris, or exploring new horizons in the tech world, remember that the sky's the limit, and the journey is always filled with surprises. Keep coding, keep gaming, and keep exploring!



**Ajay Kumar Yegireddi** is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

## Comments

### Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment \*

Name \*

Email \*

Website

Save my name, email, and website in this browser for the next time I comment.



I'm not a robot

reCAPTCHA  
[Privacy](#) - [Terms](#)[Post Comment](#)

Uncategorized

**How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point**

22 July 2025

AI

**How to Run Docker Model Runner on Ubuntu 24.04**

11 July 2025

AI, DevOps

**How to Install docker-ai on Ubuntu 24.04**

15 June 2025

## Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.



## Important Links

[Privacy Policy](#)[Terms & Conditions](#)[Contact](#)

## Resources

[Blog](#)

YouTube Channel

© 2024 · Powered by [Mr Cloud Book](#)

[Follow Us on YouTube](#)