

Name: **Đỗ Minh Trí**

Staff Code: **SD5976**

AWS Infrastructure With ArgoCD

I. Setting Up AWS Infrastructure with Terraform

We will deploy AWS infrastructure using Terraform through this repository:

https://github.com/tri-dominh/sd5976_aws_infrastructure

First, I created an IAM user for practical DevOps training: **sd5976**.



The screenshot shows the 'sd5976' IAM user details page. It includes sections for Summary, ARN, Console access (Enabled without MFA), Last console sign-in (Today), Access key 1 (AKIASSW4HOHDYUUX4GOR - Active, Used today, 8 days old), and Access key 2 (Create access key). A 'Delete' button is visible in the top right corner.

Configure this IAM user on local via **AWS CLI** with Access Key and Secret Access Key.

```
• ~/Desktop/sd5976_aws_infrastructure main ➔ aws configure
AWS Access Key ID [*****XWP5]: [REDACTED]4G0R
AWS Secret Access Key [*****1m6F]: [REDACTED]6ic5
Default region name [ap-southeast-1]:
Default output format [json]

• ~/Desktop/sd5976_aws_infrastructure main ➔ aws sts get-caller-identity
{
    "UserId": "[REDACTED]63C",
    "Account": "177629262279",
    "Arn": "arn:aws:iam::177629262279:user/sd5976"
}
```

Go to **sd5976_aws_infrastructure/terraform**, run **terraform init** to initialize the infrastructure.

```
● ~/Desktop/sd5976_aws_infrastructure/terraform ➔ main ➔ terraform init
Initializing the backend...
Initializing modules...
- ec2 in modules/ec2
- ecr_backend in modules/ecr
- ecr_frontend in modules/ecr
- eks in modules/eks
- security_groups in modules/security-groups
- vpc in modules/vpc
Initializing provider plugins...
- Finding hashicorp/tls versions matching ">~ 4.0"...
- Finding hashicorp/aws versions matching ">~ 5.70.0"...
- Finding latest version of hashicorp/local...
- Installing hashicorp/tls v4.1.0...
- Installed hashicorp/tls v4.1.0 (signed by HashiCorp)
- Installing hashicorp/aws v5.70.0...
- Installed hashicorp/aws v5.70.0 (signed by HashiCorp)
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
```

Use **terraform fmt** and **terraform validate** to format and check your code.

```
● ~/Desktop/sd5976_aws_infrastructure/terraform ➔ main ➔ terraform fmt
terraform.tfvars

● ~/Desktop/sd5976_aws_infrastructure/terraform ➔ main !1 ➔ terraform validate
Success! The configuration is valid.
```

Run **terraform plan** to preview resources that will be created.

```
Plan: 38 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ configure_kubectl          = "aws eks update-kubeconfig --region ap-southeast-1 --name cicd-pipeline-cluster"
+ docker_ip                   = (known after apply)
+ ecr_backend_repository_name = "sd5976-backend"
+ ecr_backend_repository_url  = (known after apply)
+ ecr_frontend_repository_name= "sd5976-frontend"
+ ecr_frontend_repository_url = (known after apply)
+ eks_cluster_endpoint        = (known after apply)
+ eks_cluster_name            = "cicd-pipeline-cluster"
+ jenkins_initial_password_command = (known after apply)
+ jenkins_ip                  = (known after apply)
+ jenkins_url                 = (known after apply)
+ ssh_connection_docker       = (known after apply)
+ ssh_connection_jenkins      = (known after apply)
+ vpc_id                      = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

○ ~/Desktop/sd5976_aws_infrastructure/terraform ➔ main ➔
```

Execute **terraform apply** to deploy the resources.

Type **yes** to confirm.

```
Plan: 38 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ configure_kubectl      = "aws eks update-kubeconfig --region ap-southeast-1 --name cicd-pipeline-cluster"
+ docker_ip                = (known after apply)
+ ecr_backend_repository_name = "sd5976-backend"
+ ecr_backend_repository_url = "177629262279.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-backend"
+ ecr_frontend_repository_name = "sd5976-frontend"
+ ecr_frontend_repository_url = "177629262279.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-frontend"
+ eks_cluster_endpoint     = (known after apply)
+ eks_cluster_name          = "cicd-pipeline-cluster"
+ jenkins_initial_password_command = (known after apply)
+ jenkins_ip                  = (known after apply)
+ jenkins_url                 = (known after apply)
+ ssh_connection_docker       = (known after apply)
+ ssh_connection_jenkins      = (known after apply)
+ vpc_id                      = (known after apply)

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
```

Resource creation may take around 20 minutes or more.

```
Apply complete! Resources: 38 added, 0 changed, 0 destroyed.

Outputs:

configure_kubectl = "aws eks update-kubeconfig --region ap-southeast-1 --name cicd-pipeline-cluster"
docker_ip = "47.128.237.198"
ecr_backend_repository_name = "sd5976-backend"
ecr_backend_repository_url = "177629262279.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-backend"
ecr_frontend_repository_name = "sd5976-frontend"
ecr_frontend_repository_url = "177629262279.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-frontend"
eks_cluster_endpoint = "https://9A770A8B363B32838F392C1018289B7E.gr7.ap-southeast-1.eks.amazonaws.com"
eks_cluster_name = "cicd-pipeline-cluster"
jenkins_initial_password_command = "ssh -i cicd-pipeline-key.pem ubuntu@13.228.28.131 'sudo cat /var/lib/jenkins/secrets/initialAdminPassword'"
jenkins_ip = "13.228.28.131"
jenkins_url = "http://13.228.28.131:8080"
ssh_connection_docker = "ssh -i cicd-pipeline-key.pem ubuntu@47.128.237.198"
ssh_connection_jenkins = "ssh -i cicd-pipeline-key.pem ubuntu@13.228.28.131"
vpc_id = "vpc-0184de0dfb2c42624"

○ ~/Desktop/sd5976_aws_infrastructure/terraform ➜ main ➔ ]
```

After applying, you can see the following resources on AWS:

VPC

Resources by Region		
You are using the following Amazon VPC resources		
VPCs ▶ See all regions	Singapore 2	NAT Gateways ▶ See all regions
Subnets ▶ See all regions	Singapore 7	VPC Peering Connections ▶ See all regions
Route Tables ▶ See all regions	Singapore 4	Network ACLs ▶ See all regions
Internet Gateways ▶ See all regions	Singapore 2	Security Groups ▶ See all regions
Egress-only Internet Gateways ▶ See all regions	Singapore 0	Customer Gateways ▶ See all regions
DHCP option sets ▶ See all regions	Singapore 1	Virtual Private Gateways ▶ See all regions
Endpoints ▶ See all regions	Singapore 0	Site-to-Site VPN Connections ▶ See all regions
Instance Connect Endpoints ▶ See all regions	Singapore 0	Running Instances ▶ See all regions
Endpoint Services ▶ See all regions	Singapore 0	

The screenshot shows the AWS VPC dashboard with the following details:

- Left sidebar:** Includes sections for VPC dashboard, Virtual private cloud (with Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, and Route servers), Security (Network ACLs, Security groups), and PrivateLink and Lattice.
- Top bar:** Shows the URL https://ap-southeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-southeast-1#vpcs, the AWS logo, a search bar, and various navigation icons.
- Main content area:**
 - Your VPCs (2) Info:** A table listing two VPCs:

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
cldc-pipeline-vpc	vpc-0184de0dfb2c42624	Available	Off	10.0.0.0/16	-	dopt-00f00fdb9c3280f8e	rtb-0d7613a9b048
-	vpc-d55d38068b5fc353c	Available	Off	172.31.0.0/16	-	dopt-00f00fdb9c3280f8e	rtb-0d5f821e3f31
 - Select a VPC above:** A placeholder text indicating which VPC to select.

EC2

The screenshot shows the AWS EC2 Instances page. The left sidebar navigation includes: Dashboard, AWS Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing. The main content area displays a table titled 'Instances (4) Info' with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 address, and Elastic IP. The instances listed are: i-0e95090e556111d12 (Running, t3.small, 3/3 checks passed, View alarms, ap-southeast-1a, ec2-13-228-28-131.ap..., 13.228.28.131), i-0df7e2ca8a201da9e (Running, c7i-flex.large, 3/3 checks passed, View alarms, ap-southeast-1a, ec2-13-228-28-131.ap..., 13.228.28.131), i-00b30d2c405854abc (Running, t3.small, 3/3 checks passed, View alarms, ap-southeast-1b, -), and i-050419e902f16505a (Running, t3.small, 3/3 checks passed, View alarms, ap-southeast-1b, ec2-47-128-237-198.ap..., 47.128.237.198). Buttons at the top right include 'Connect', 'Instance state', 'Actions', and 'Launch instances'. The bottom right corner shows copyright information: © 2025, Amazon Web Services, Inc. or its affiliates.

EKS

The screenshot shows the AWS EKS Clusters page. The left sidebar navigation includes: Dashboard (New), Clusters (selected), Settings (Dashboard settings, Console settings), Amazon EKS Anywhere (Enterprise Subscriptions), and Related services (Amazon ECR, AWS Batch). The main content area displays a table titled 'Clusters (1) Info' with columns: Cluster name, Status, Kubernetes version, Support period, Upgrade policy, Created, and Provider. The cluster listed is 'cicd-pipeline-cluster' (Active, 1.31, Upgrade now, Standard support until November 26, 2025, Extended support, 24 minutes ago, EKS). Buttons at the top right include 'Delete' and 'Create cluster'. The bottom right corner shows copyright information: © 2025, Amazon Web Services, Inc. or its affiliates.

https://ap-southeast-1.console.aws.amazon.com/eks/clusters/cicd-pipeline-cluster?region=ap-southeast-1

Amazon Elastic Kubernetes Service > Clusters > cicd-pipeline-cluster

cicd-pipeline-cluster

End of standard support for Kubernetes version 1.31 is November 26, 2025.

Cluster info

Status: Active	Kubernetes version: 1.31	Support period: Standard support until November 26, 2025	Provider: EKS
Cluster health: 0	Upgrade insights: 6	Node health issues: 0	

Overview | Resources | Compute | Networking | Add-ons | Access | Observability | Update history | Tags

Details

API server endpoint: https://9A770A8B363B32838F392C101828987E.gr7.ap-southeast-1.eks.amazonaws.com	OpenID Connect provider URL: https://oidc.eks.ap-southeast-1.amazonaws.com/id/9A770A8B363B32838F392C101828987E
Certificate authority: LSQhLS1CRU4UTIBDRVJUSUZ400FUR50tL50iCk1j5URCVENDQWUy20f35UJB20U1JaajlQnK2SV15RF725ktbWkd0mHOQVFTTEJRQXgGVEVUTUJFR0ExvUUKQXhNS2EzVmIaWEp1WhsbgN5Q	Cluster IAM role ARN: arn:aws:iam::17762962279:role/cicd-pipeline-eks-cluster-role
EKS Auto Mode: Disabled	Created: 25 minutes ago
	Cluster ARN: arn:aws:eks:ap-southeast-1:17762962279:cluster/cicd-pipeline-cluster
	Platform version: eks.44

EKS Auto Mode

EKS automates routine cluster tasks for compute, storage, and networking to meet application compute needs.

EKS Auto Mode: Disabled

Manage

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

II. Access Jenkins

Terraform has already created an EC2 instance with Jenkins installed. Access Jenkins at:

<http://13.228.28.131:8080/>

The screenshot shows the AWS EC2 Instances page. The instance summary for the Jenkins instance (i-0df7e2ca8a201da9e) is displayed. Key details include:

- Public IPv4 address:** 13.228.28.131 (highlighted with a red box)
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-10-0-3-185.ap-southeast-1.compute.internal
- Instance type:** c7i-flex-large
- VPC ID:** vpc-0184de0dfb2c42624 (cicd-pipeline-vpc)
- Subnet ID:** subnet-09e5afbec835ccaf5 (cicd-pipeline-public-1)
- Instance ARN:** arn:aws:ec2:ap-southeast-1:177629262279:instance/i-0df7e2ca8a201da9e

The left sidebar shows the navigation menu for EC2, including Dashboard, AWS Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, and Load Balancing.

The screenshot shows the Jenkins 'Unlock Jenkins' setup page. The page title is 'Getting Started' and the main heading is 'Unlock Jenkins'. It instructs the user to copy the password from either the log or the file /var/lib/jenkins/secrets/initialAdminPassword. A text input field labeled 'Administrator password' is provided for pasting the password. A 'Continue' button is located at the bottom right of the form.

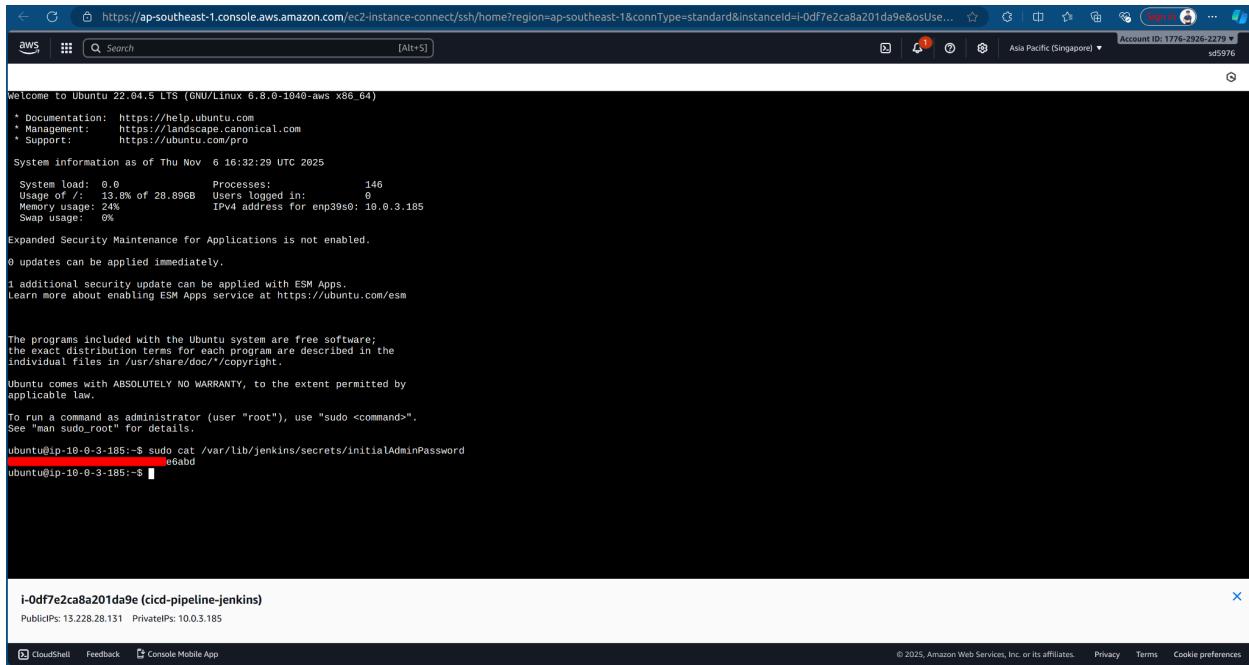
To get the initialAdminPassword, we need to connect to the EC2 instance **cicd-pipeline-jenkins**.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like Dashboard, AWS Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, and Load Balancing. The main content area displays the instance summary for 'i-0df7e2ca8a201da9e (cicd-pipeline-jenkins)'. It includes details such as Instance ID (i-0df7e2ca8a201da9e), Public IPv4 address (13.228.28.131), Instance state (Running), Hostname type (IP name: ip-10-0-3-185.ap-southeast-1.compute.internal), Auto-assigned IP address (13.228.28.131 [Public IP]), IAM Role (c7n-flex-large), IMDSv2 (Optional), Operator (None), and various network and storage details. At the top right, there's a 'Connect' button, which is highlighted with a red box.

The screenshot shows the 'Connect to instance' dialog box. It has tabs for EC2 Instance Connect, Session Manager, SSH client, and EC2 serial console. The EC2 Instance Connect tab is active. It shows the instance ID (i-0df7e2ca8a201da9e (cicd-pipeline-jenkins)) and two connection type options: 'Connect using a Public IP' (selected) and 'Connect using a Private IP'. Below these are fields for 'Public IPv4 address' (13.228.28.131) and 'Username' (ubuntu). A note at the bottom states: 'Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' At the bottom right, there are 'Cancel' and 'Connect' buttons, with 'Connect' highlighted with a red box.

Retrieve the initial admin password:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



```
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1049-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Thu Nov  6 16:32:29 UTC 2025
System load: 0.0          Processes:           146
Usage of /: 13.8% of 28.89GB   Users logged in: 0
Memory usage: 24%          IPv4 address for enp39s0: 10.0.3.185
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

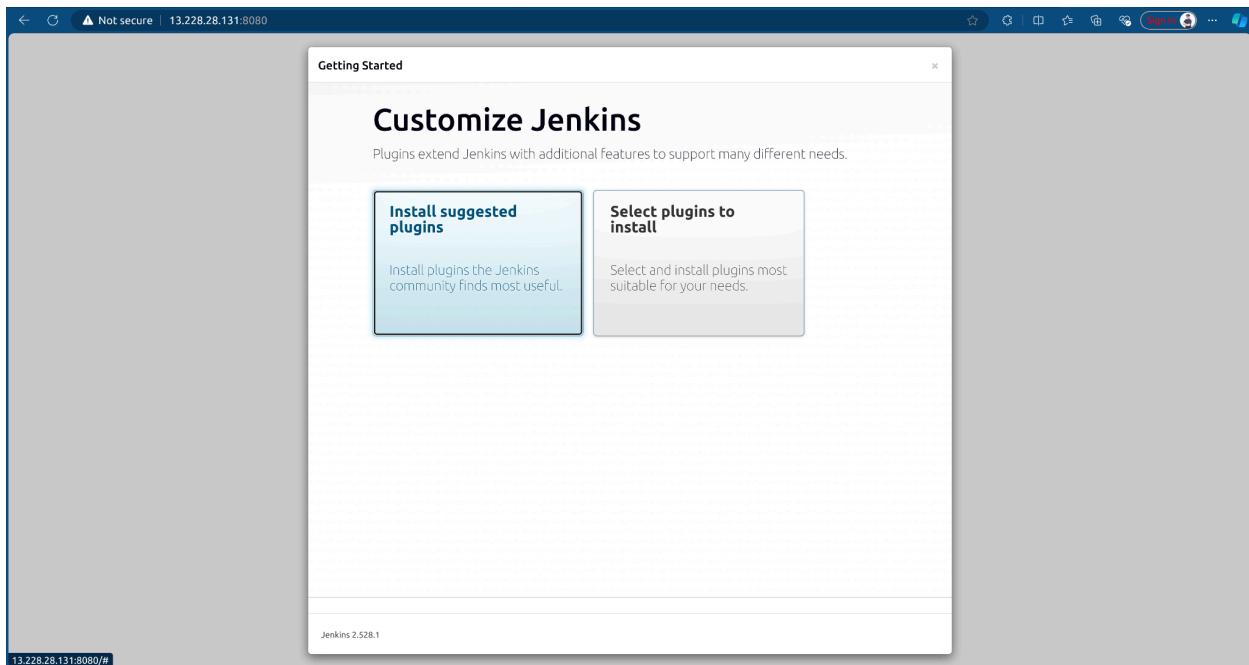
ubuntu@ip-10-0-3-185:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e6abd
ubuntu@ip-10-0-3-185:~$ 
```

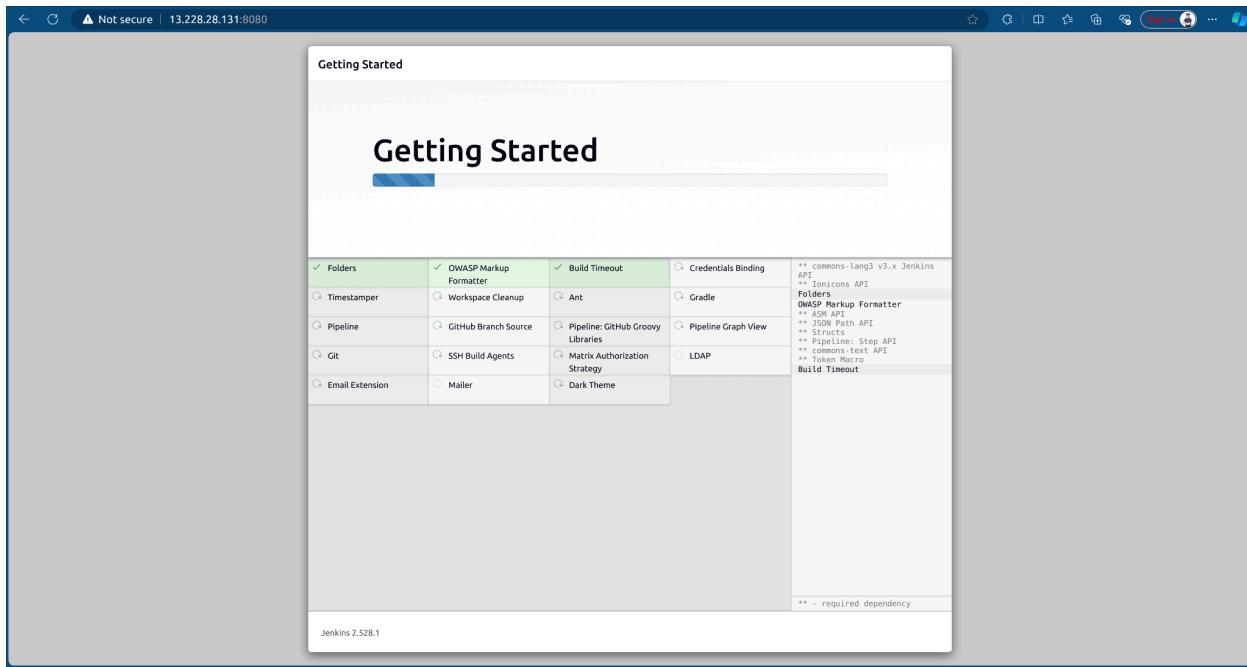
i-0df7e2ca8a201da9e (cicd-pipeline-jenkins)
Public IPs: 13.228.28.131 Private IPs: 10.0.3.185

CloudShell Feedback Console Mobile App

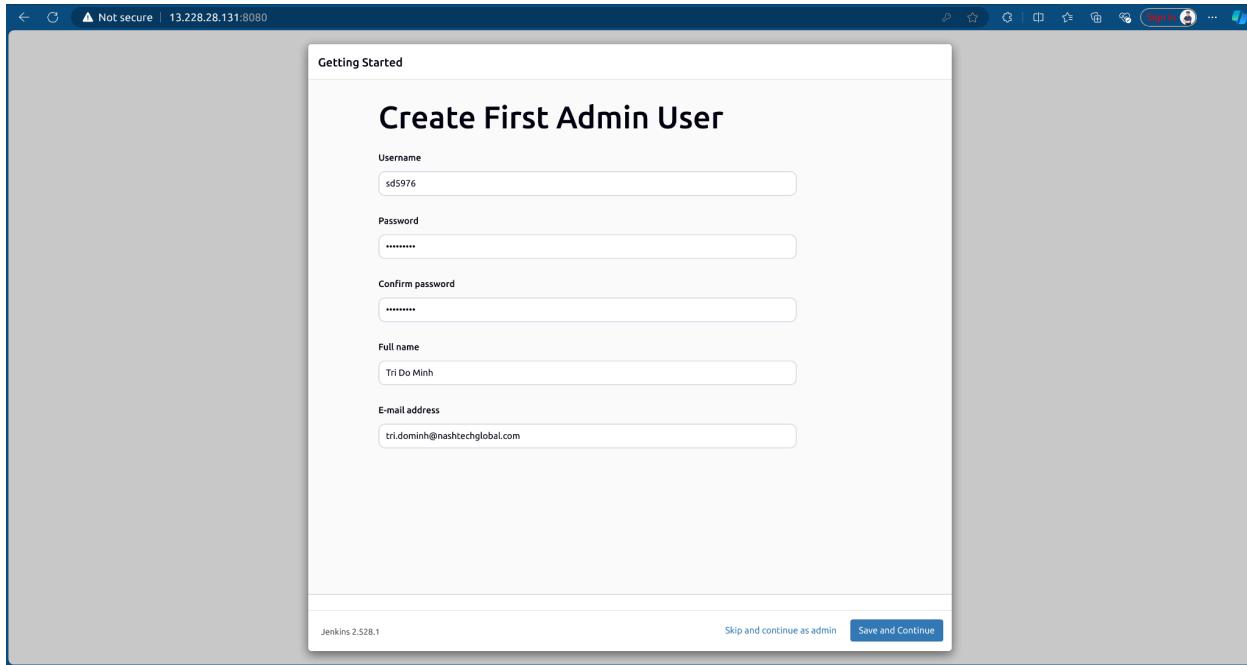
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Copy the password and paste it into Jenkins UI, then select **Install suggested plugins**.





After installation, create an Admin User to log in to the dashboard.



The screenshot shows the Jenkins home page at 13.228.28.131:8080. The top navigation bar includes links for 'Sign in' and 'Help'. The main content area features a 'Welcome to Jenkins!' message with a sub-instruction: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this, there are sections for 'Start building your software project' (with a 'Create a job' button) and 'Set up a distributed build' (with 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds' buttons). On the left, there are two dropdown menus: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (9/2). The bottom right corner displays 'REST API' and 'Jenkins 2.528.1'.

To run CI/CD, we need to install some plugins:

- Docker
- Docker Pipeline
- Pipeline: AWS Steps

Go to **Manage Jenkins → Plugins → Available plugins** tab.

The screenshot shows the 'Available plugins' tab in the Jenkins plugin manager. The left sidebar has links for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area lists available plugins with columns for 'Install', 'Name', 'Released', and 'Health'. Two plugins are selected for installation: 'Docker' (version 1274.vc0203df2e74) and 'Docker Pipeline' (version 634.vedc7242b_eda_7). Both have a green circular health status icon with the number 96. Other listed plugins include 'Pipeline Graph Analysis', 'PAM Authentication', 'JavaMail API', 'Command Agent Launcher', 'Oracle Java SE Development Kit Installer', 'SSH server', 'Pipeline: REST API', and 'User interface'. A red box highlights the 'Install' button at the top right of the table.

Restart Jenkins after plugin installation.

Not secure | 13.228.28.131:8080/manage/pluginManager/updates/

Jenkins / Manage Jenkins / Plugins

Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Plugin	Status
Commons Compress API	Success
Docker API	Success
Docker	Success
Docker Pipeline	Success
Loading plugin extensions	Success
Amazon Web Services SDK : Minimal	Success
Amazon Web Services SDK : SQS	Success
Amazon Web Services SDK : SNS	Success
Amazon Web Services SDK : Api Gateway	Success
Amazon Web Services SDK 2 : Core	Success
Amazon Web Services SDK 2 : EC2	Success
Amazon Web Services SDK : EC2	Success
AWS Credentials	Success
Amazon Web Services SDK : CloudFormation	Success
Amazon Web Services SDK : Elastic Beanstalk	Success
Amazon Web Services SDK : Elastic Load Balancing V2	Success
Amazon Web Services SDK : IAM	Success
Amazon Web Services SDK : ECR	Success
Amazon Web Services SDK : CloudFront	Success
Amazon Web Services SDK : Lambda	Success
Amazon Web Services SDK : Organizations	Success
Amazon Web Services SDK : CodeDeploy	Success
Pipeline: AWS Steps	Success
Loading plugin extensions	Success

→ Go back to the top page
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.528.1

Not secure | 13.228.28.131:8080/manage/pluginManager/updates/

Jenkins is restarting

Your browser will reload automatically when Jenkins is ready.

Safe Restart
Builds on agents can usually continue.

III. Deploying the MSA Application to EKS with Jenkins CI/CD

Prepare the MSA repository containing *frontend* and *backend*:

https://github.com/tri-dominh/sd5976_msa

The screenshot shows the GitHub repository page for 'sd5976_msa'. The main branch has one commit from 'tri-dominh' titled 'Init project' made at 21680f8 1 minute ago. The commit includes several files: 'backend', 'frontend', 'react-express-mongodb', '.gitignore', 'README.md', 'compose.yaml', and 'output.png'. All files are listed as 'Init project'.

Each source (*frontend/backend*) should have a **Jenkinsfile** defining agent and stages: install, build, validate, test, and push to ECR.

https://github.com/tri-dominh/sd5976_msa/blob/main/backend/Jenkinsfile

https://github.com/tri-dominh/sd5976_msa/blob/main/frontend/Jenkinsfile

The screenshot shows the GitHub repository page for 'sd5976_msa' with the 'main' branch selected. The 'backend' directory is expanded, showing its contents. Inside 'backend', there is a 'Jenkinsfile' highlighted with a red box. The right pane displays the commit history for this file, which is identical to the main branch's commit history: 'Init project' at 21680f8 1 minute ago. Below the commit table, a snippet of the Jenkinsfile is shown:

```
Snippet of backend(Node.js) DockerFile
You will find this Dockerfile file in the root directory of the project.
FROM node:13.13.0-stretch-slim
```

Next, we need to configure Jenkins to access GitHub and AWS resources.

Create a GitHub Access Token for Jenkins to access the MSA repository.

Go to **Settings** → **Developer settings** → **Personal access tokens** → **Tokens (classic)** → **Generate new token (classic)** → Enter token name and select scopes to allow Jenkins access to GitHub → Create.

The screenshot shows the GitHub profile settings page at <https://github.com/settings/profile>. The 'Developer settings' section is highlighted with a red box. It contains fields for 'Company' (NashTech), 'Location', and 'Display current local time'. There's also an 'ORCID ID' section and a 'Connect your ORCID ID' button. Below these are optional fields for hiding activity and showing achievements. At the bottom are 'Update profile' and 'Contributions & activity' sections.

The screenshot shows the GitHub developer settings page at <https://github.com/settings/tokens>. The 'Personal access tokens (classic)' section is highlighted with a red box. It shows a list of generated tokens and a button to 'Generate new token'. A sub-section titled 'Generate new token (classic)' is also highlighted with a red box, describing it as 'Fine-grained, repo-scoped' and 'For general use'. The left sidebar includes 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (which is currently selected). The bottom navigation bar includes links for Terms, Privacy, Security, Status, Community, Docs, Contact, Manage cookies, and Do not share my personal information.

<https://github.com/settings/tokens/new>

Settings / Developer Settings

GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note: sd5976-practical-devops

What's this token for?

Expiration: 30 days (Dec 07, 2025)

The token will expire on the selected date.

Select scopes: Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repository	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys

<https://github.com/settings/tokens/new>

<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> scm:enterprise	Provisioning of users and groups via SCIM

The screenshot shows the GitHub Developer Settings page at <https://github.com/settings/tokens>. The left sidebar is titled "Developer Settings" and includes options for "GitHub Apps", "OAuth Apps", "Personal access tokens", "Fine-grained tokens", and "Tokens (classic)". The "Tokens (classic)" option is selected. The main area is titled "Personal access tokens (classic)" and contains a button "Generate new token". Below it, a table lists a single token:

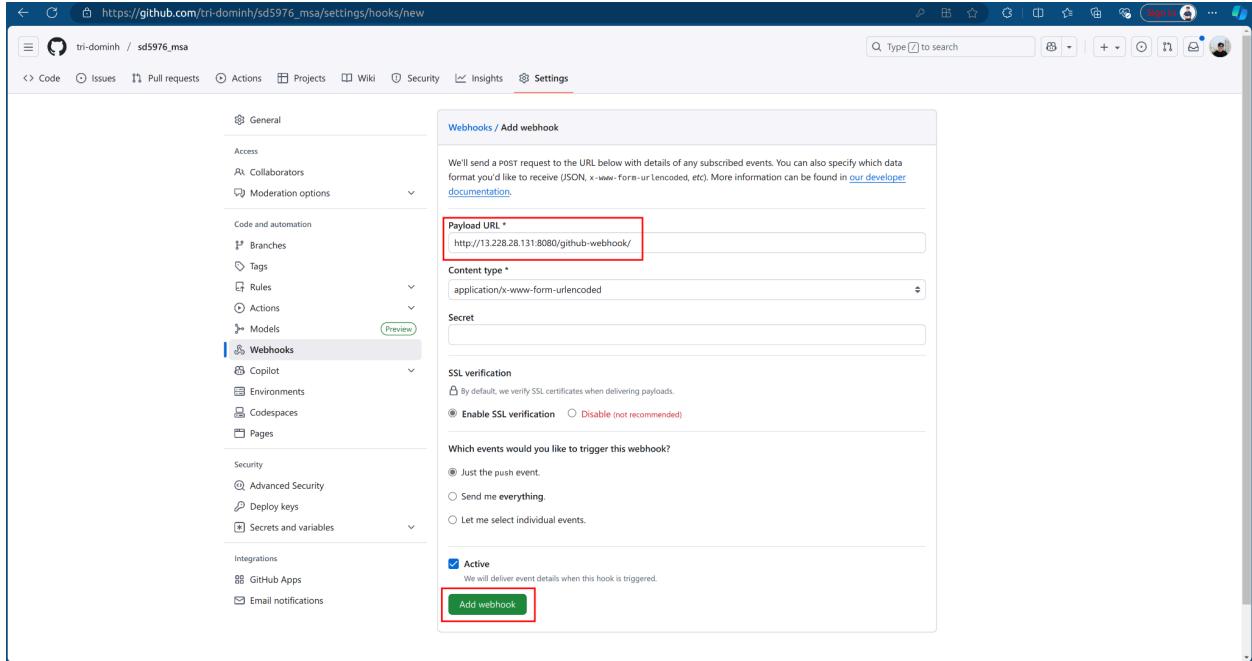
sd5976-practical-devops	admin:org, admin:org_hook, admin:public_key, admin:repo_hook, repo, workflow	Never used	Delete
Expires on Sun, Dec 7 2025.			

A note below the table states: "Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication."

At the bottom of the page, there is a footer with links: © 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information.

This screenshot is identical to the one above, but it includes a prominent blue warning message box at the top of the token list area: "Make sure to copy your personal access token now. You won't be able to see it again!"

Then, we create a Github webhook to trigger Jenkins whenever we push new code. In the **sd5976-msa** repo, go to **Settings → Webhooks → Add webhook**, enter the Jenkins URL with */github-webhook/*



After that, we create the credentials in Jenkins which allow Jenkins access to Resources. Go to **Jenkins Credentials → Click on (global) → Add Credentials (Secret text kind)**.

- AWS Account ID, AWS Access Key ID, AWS Secret Access Key.
- GitHub Token.

Not secure | 13.228.28.131:8080/manage/ Jenkins / Manage Jenkins

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

System Configuration

- System: Configure global settings and paths.
- Tools: Configure tools, their locations and automatic installers.
- Plugins: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Docker: Plugin for launching build Agents as Docker containers.

Clouds: Add, remove, and configure cloud instances to provision agents on-demand.

Appearance: Configure the look and feel of Jenkins.

Security

- Security: Secure Jenkins; define who is allowed to access/use the system.
- Credentials: Configure credentials (highlighted with a red box).
- Credential Providers: Configure the credential providers and types.
- Users: Create/delete/modify users that can log in to this Jenkins.

Status Information

- System Information: Displays various environmental information to assist trouble-shooting.
- System Log: System log captures output from java.util.logging output related to Jenkins.
- Load Statistics: Check your resource utilization and see if you need more computers for your builds.
- About Jenkins: See the version and license information.

Troubleshooting

- Manage Old Data: Scrub configuration files to remove remnants from old plugins and earlier versions.

Set up agent Set up cloud Dismiss

Search settings

Not secure | 13.228.28.131:8080/manage/credentials/ Jenkins / Manage Jenkins / Credentials

Credentials

T	P	Store	Domain	ID	Name

Stores scoped to Jenkins

P	Store	Icon: S M L
	System	
	Domains (global)	

REST API Jenkins 2.528.1

Not secure | 13.228.28.131:8080/manage/credentials/store/system/domain/_/ Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrest...)

Global credentials (unrestricted)

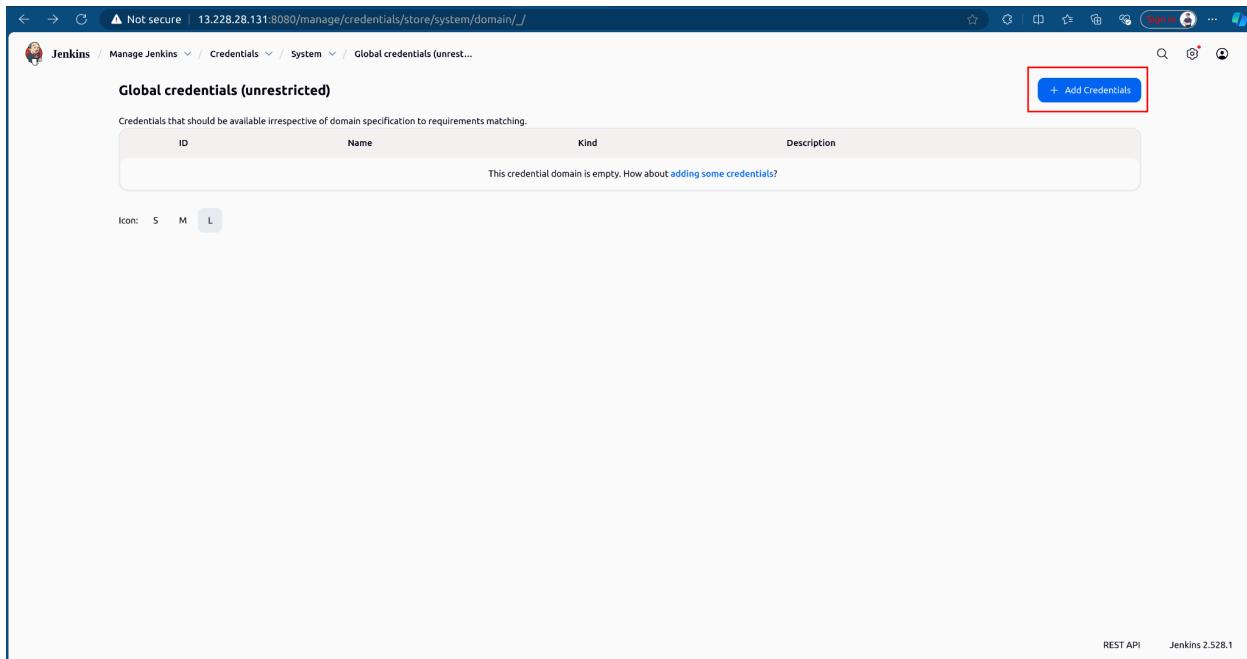
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

Icon: S M L

+ Add Credentials

REST API Jenkins 2.528.1



Not secure | 13.228.28.131:8080/manage/credentials/store/system/domain/_/ Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrest...)

Global credentials (unrestricted)

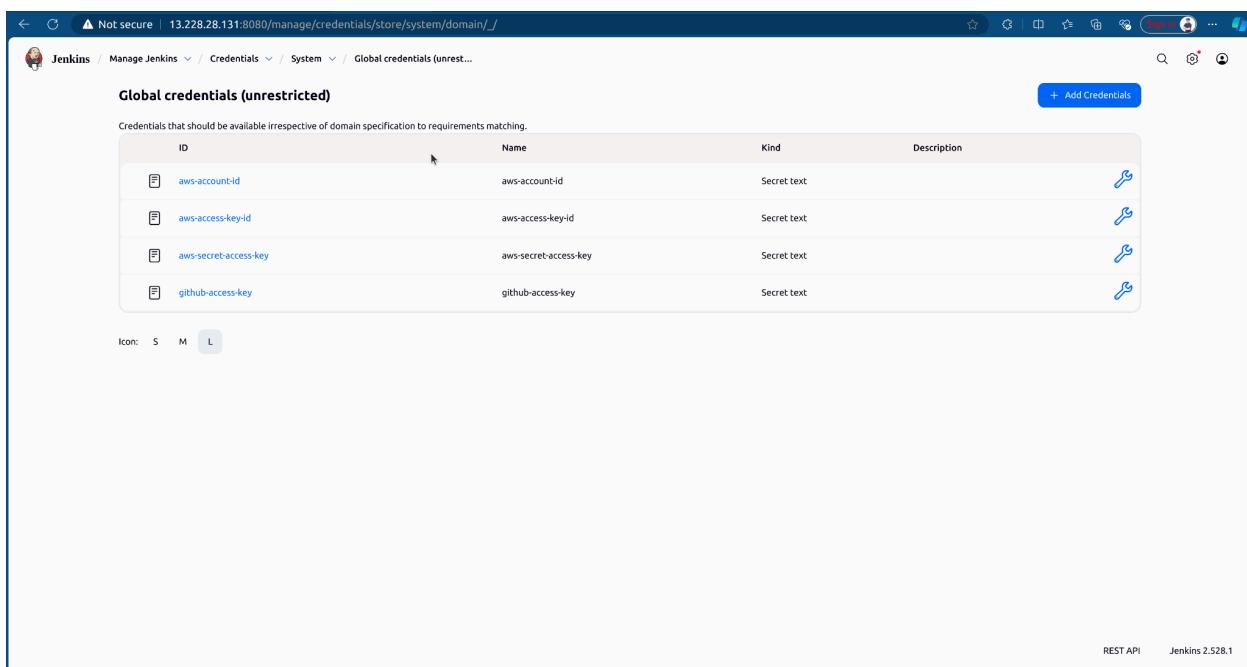
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
aws-account-id	aws-account-id	Secret text	
aws-access-key-id	aws-access-key-id	Secret text	
aws-secret-access-key	aws-secret-access-key	Secret text	
github-access-key	github-access-key	Secret text	

Icon: S M L

+ Add Credentials

REST API Jenkins 2.528.1



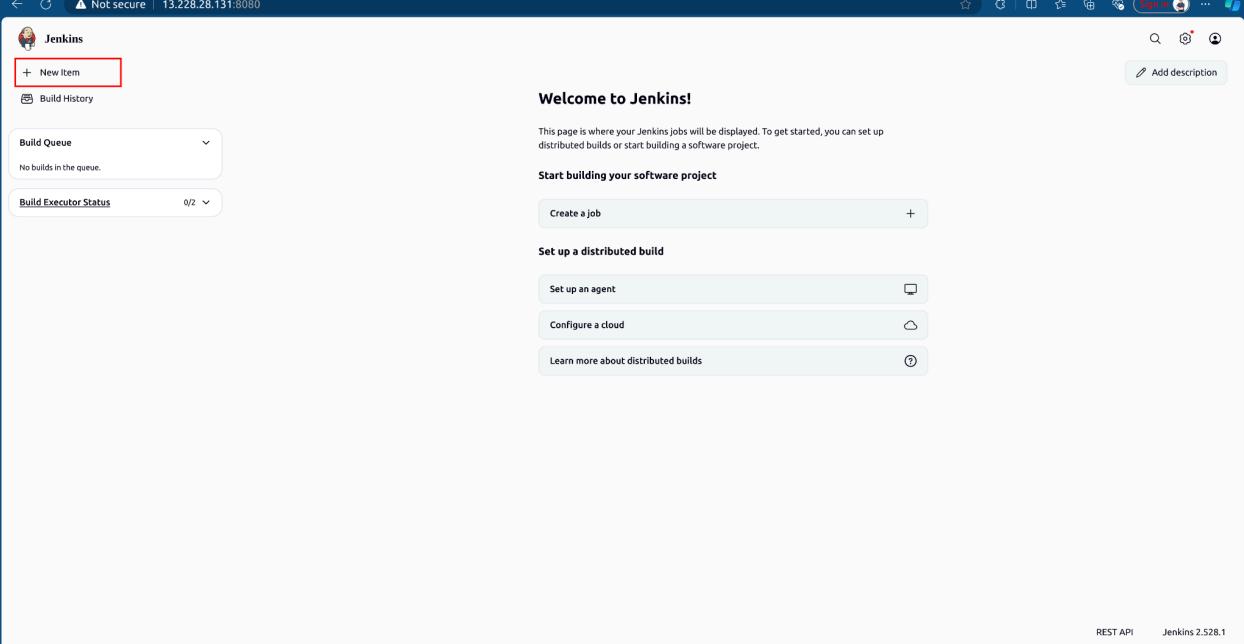
Add GitHub Server for Jenkins, go to **Settings** → **System** → In **GitHub** section → **Add GitHub Server** → Enter name, select credentials **github-access-key** and tick **Manage hooks** → Save.

The screenshot shows the Jenkins 'Manage Jenkins' interface at the URL <http://13.228.28.131:8080/manage/>. The 'System Configuration' section is highlighted with a red box around the 'System' icon. Other sections shown include Tools, Clouds, Plugins, Appearance, Nodes, Security, Credentials, Credential Providers, and Users. The 'System' section contains links for 'System Information', 'System Log', 'Load Statistics', and 'About Jenkins'.

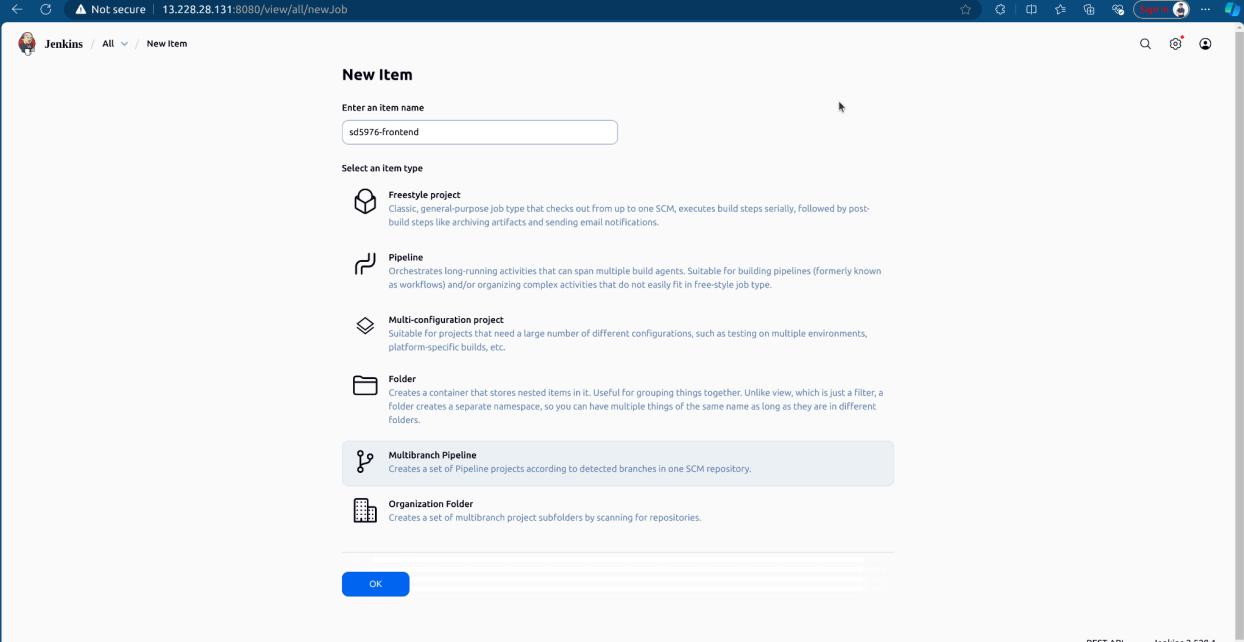
The screenshot shows the Jenkins 'Manage Jenkins' interface at the URL <http://13.228.28.131:8080/manage/configure>. The 'GitHub' section is expanded, showing the 'GitHub Servers' configuration. A new server named 'sd5976-github-msa' is being added. The 'Name' field is highlighted with a red box. The 'API URL' field contains 'https://api.github.com'. The 'Credentials' dropdown is set to 'github-access-key', which is also highlighted with a red box. The 'Manage hooks' checkbox is checked and highlighted with a red box. Below the form are 'Advanced' and '+ Add GitHub Server' buttons. At the bottom, there are 'GitHub API usage' buttons for 'Save' and 'Apply', with 'Save' highlighted with a red box.

IV. Creating CI jobs for MSA Application

First, we build a job for the *Frontend*, click **New item** → Enter name and select **Multibranch pipeline job**.



The screenshot shows the Jenkins dashboard at the URL <http://13.228.28.131:8080>. The 'New Item' button is highlighted with a red box. The page includes sections for 'Welcome to Jenkins!', 'Start building your software project', and 'Set up a distributed build'.



The screenshot shows the 'New Item' dialog at the URL <http://13.228.28.131:8080/view/all/newJob>. The 'Multibranch Pipeline' option is selected and highlighted with a blue box. The dialog also shows fields for 'Enter an item name' (sd5976-frontend) and 'Select an item type'.

Configure Jenkins job to point to the *Frontend* branch and Jenkinsfile of *Frontend*.

The screenshot shows the Jenkins job configuration page for 'sd5976-frontend'. The 'General' section is active, displaying the 'Display Name' field with 'sd5976-frontend' and the 'Enabled' switch turned on. The 'Branch Sources' section contains a 'Git' configuration with a 'Project Repository' set to 'https://github.com/tri-dominh/sd5976_msa.git'. The 'Build Configuration' section is visible at the bottom.

The screenshot shows the Jenkins job configuration page for 'sd5976-frontend'. The 'Build Configuration' section is active, displaying the 'Mode' dropdown set to 'by Jenkinsfile' and the 'Script Path' input field containing 'frontend/Jenkinsfile'. The 'Orphaned Item Strategy' section is also visible at the bottom.

After creating, the job will run Jenkins based on the Jenkinsfile and source.

Jenkins / sd5976-frontend / main / #1 / Pipeline Overview

#1

21680f8 Started 2 min 4 sec ago Queued 6.7 sec Took 2 min 4 sec and counting

Graph

Start Checkout SCM Checkout Install Dependencies Lint Test Build Application Build Docker Image Login to AWS ECR Push to ECR Post Actions End

Post Actions

Checkout SCM 1.0s
Checkout 0.73s
Install Dependencies 1m 20s
Lint 34ms
Test 30ms
Build Application 9.3s
Build Docker Image 1m 54s
Login to AWS ECR 3.0s
Push to ECR 30s
Post Actions 0.36s

Cleaning up...
docker rmi \${AWS_ACCOUNT_ID}.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-frontend:1 || true docker rmi \${AWS_ACCOUNT_ID}.dkr.ecr.ap-southeast-1.amazonaws.co...
Frontend Pipeline completed successfully!
Image: \${AWS_ACCOUNT_ID}.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-frontend:1

Warning: A secret was passed to "echo" using Groovy String Interpolation, which is insecure.
Affected argument(s) used the following variable(s): \${AWS_ACCOUNT_ID}
See <https://jenkins.io/redirect/groovy-string-interpolation> for details.
Image: ****.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-frontend:1

0.36s Started 19s ago Jenkins Jenkins 2.528.1

CI will run and push the *Frontend* image to ECR.

https://ap-southeast-1.console.aws.amazon.com/ecr/repositories/private/177629262279/sd5976-frontend?region=ap-southeast-1

Amazon ECR > Private registry > Repositories > sd5976-frontend

Amazon Elastic Container Registry

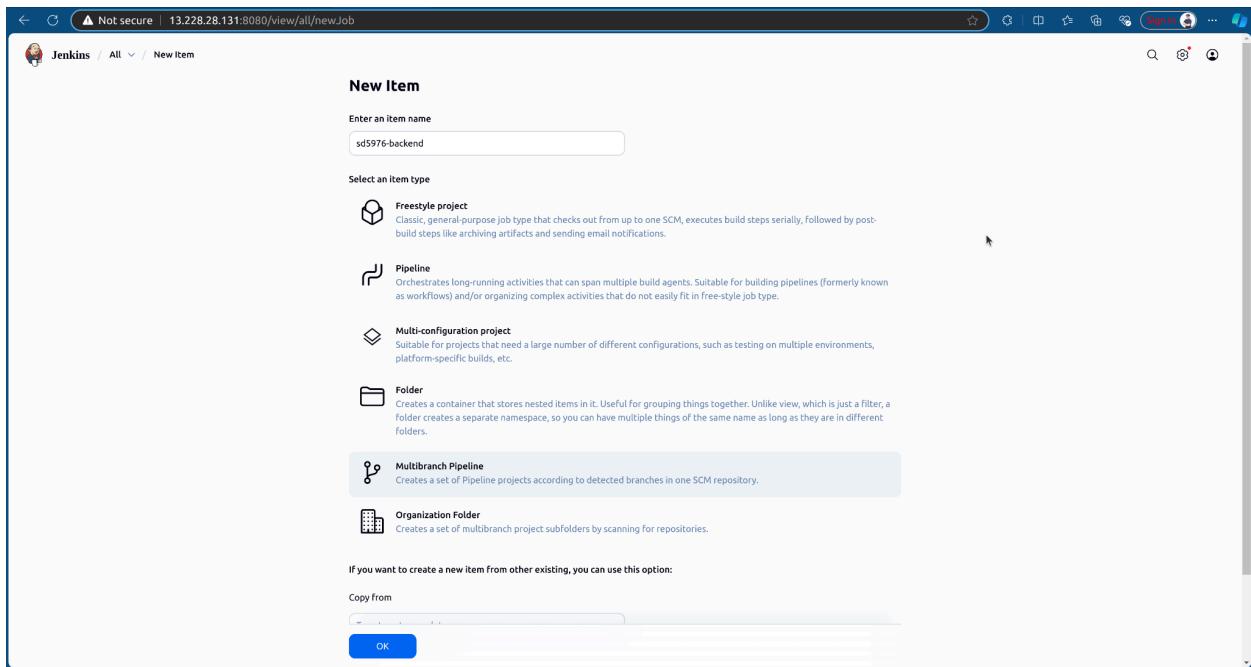
Private registry

Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest_1	Image	November 07, 2025, 00:42:22 (UTC+07)	486.73	Copy URI	sha256:a60aedf80c394e4a90016a70e9c612...	-

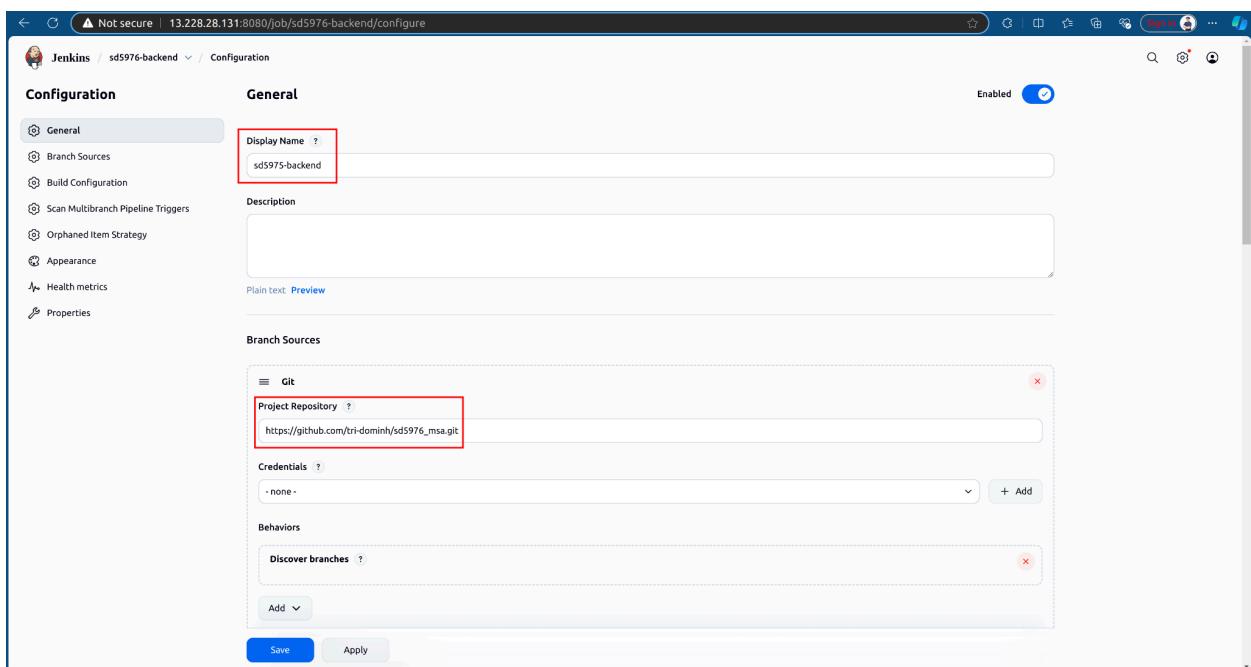
CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Next, we will create a similar job for the *Backend*, click **New item** → Enter name and select **Multibranch pipeline** job.



The screenshot shows the Jenkins 'New Item' creation interface. In the 'Enter an item name' field, the text 'sd5976-backend' is entered. Under 'Select an item type', the 'Multibranch Pipeline' option is selected, indicated by a blue background. Other options shown include 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', and 'Organization Folder'. A note at the bottom left says 'If you want to create a new item from other existing, you can use this option:' followed by a 'Copy from' dropdown and 'OK' and 'Cancel' buttons.

Configure Jenkins job to point to *Backend* branch and Jenkinsfile of *Backend*.



The screenshot shows the Jenkins configuration page for the 'sd5976-backend' job. The 'General' tab is active. The 'Display Name' field is set to 'sd5976-backend'. Under 'Branch Sources', a 'Git' section is expanded, showing a 'Project Repository' field with the URL 'https://github.com/tri-dominh/sd5976_msa.git'. The 'Credentials' field is set to '-none-' and has a '+ Add' button. The 'Behaviors' section includes a 'Discover branches' field with an 'Add' button. At the bottom are 'Save' and 'Apply' buttons.

Jenkins / sd5976-backend / Configuration

Build Configuration

Mode: by Jenkinsfile

Script Path: backend/Jenkinsfile

Scan Multibranch Pipeline Triggers

Periodically if not otherwise run:

Orphaned Item Strategy

Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.

Abort builds: ?

Discard old items

Days to keep old items: if not empty, old items are only kept up to this number of days

Max # of old items to keep: if not empty, only up to this number of old items are kept

Save Apply

After creating, the job will run Jenkins based on the Jenkinsfile and source.

Jenkins / sd5975-backend / main / #1 / Pipeline Overview

#1

21680FB Started 5.1 sec ago Queued 8.1 sec Took 5.1 sec and counting

Rerun

Graph

Start → Checkout SCM → Checkout → Install Dependencies → Lint → Test → Build Docker Image → Login to AWS ECR → Push to ECR → Post Actions → End

Post Actions

Step	Duration	Log
Checkout SCM	0.92s	Cleaning up...
Checkout	0.63s	docker rmi \${AWS_ACCOUNT_ID}.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-backend:1 true docker rmi \${AWS_ACCOUNT_ID}.dkr.ecr.ap-southeast-1.amazonaws.com...
Install Dependencies	13s	Backend Pipeline completed successfully!
Lint	43ms	Image: \${AWS_ACCOUNT_ID}.dkr.ecr.ap-southeast-1.amazonaws.com/sd5976-backend:1
Test	43ms	Warning: A secret was passed to "echo" using Groovy String interpolation, which is insecure. Affected argument(s) used the following variable(s): [AWS_ACCOUNT_ID] See https://jenkins.io/redirect/groovy-string-interpolation for details.
Build Docker Image	24s	
Login to AWS ECR	1.0s	
Push to ECR	7.8s	
Post Actions	0.38s	

Jenkins 2.528.1

CI will run and push the *Backend* image to ECR.

The screenshot shows the AWS ECR console with the URL <https://ap-southeast-1.console.aws.amazon.com/ecr/repositories/private/177629262279/sd5976-backend?region=ap-southeast-1>. The left sidebar shows the navigation path: AWS > Amazon ECR > Private registry > Repositories > sd5976-backend. The main content area displays a table titled 'Images (1)'. The table has columns: Image tag, Artifact type, Pushed at, Size (MB), Image URI, and Digest. One row is shown: '1, latest' (Image), November 07, 2025, 00:50:40 (UTC+07), 89.78, [Copy URI](#), and [sha256:25ae2677816fe9cd59599c01257d8...](#). There are also buttons for 'Delete', 'Details', 'Scan', and 'View push commands'.

V. Deploy with ArgoCD

Refer to source code:

https://github.com/tri-dominh/sd5976_devops_cicd_deployment/tree/main/argocd-deployment

Before starting, you need to install the ArgoCD CLI on your local machine.

Connect to your EKS cluster and create a namespace for ArgoCD:

kubectl create namespace argocd

```
~/Desktop/sd5976_aws_infrastructure | main ➔ kubectl create namespace argocd
namespace/argocd created
```

```
~/Desktop/sd5976_aws_infrastructure | main ➔ [ ]
```

Install ArgoCD to the **argocd** namespace in EKS:

kubectl apply -n argocd -f

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

```
~/De/sd5976_aws_infrastructure | main ➔ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-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-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrole.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-applicationset-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
secret/argocd-notifications-secret created
secret/argocd-secret created
service/argocd-applicationset-controller created
service/argocd-dex-server created
service/argocd-metrics created
service/argocd-notifications-controller-metrics created
service/argocd-redis created
service/argocd-repo-server created
service/argocd-server created
service/argocd-server-metrics created
deployment.apps/argocd-applicationset-controller created
deployment.apps/argocd-dex-server created
deployment.apps/argocd-notifications-controller created
deployment.apps/argocd-redis created
deployment.apps/argocd-repo-server created
deployment.apps/argocd-server created
statefulset.apps/argocd-application-controller created
networkpolicy.networking.k8s.io/argocd-application-controller-network-policy created
```

Wait for a few minutes while Kubernetes creates the pods, then check their status:

kubectl get pods -n argocd

NAME	READY	STATUS	RESTARTS	AGE
argocd-application-controller-0	1/1	Running	0	50s
argocd-applicationset-controller-5c8d57cb-twmxx	1/1	Running	0	51s
argocd-dex-server-77978c8bbc-8r8n7	1/1	Running	0	51s
argocd-notifications-controller-59db9f4c79-ddj84	1/1	Running	0	51s
argocd-redis-c9bdd6f47-rdzbn	1/1	Running	0	51s
argocd-repo-server-6c499f5448-8g5l8	1/1	Running	0	51s
argocd-server-58b9cbff8d-tq29t	1/1	Running	0	51s

We need to get the initial password and decode it from the base64-encoded secret:

```
kubectl -n argo cd get secret argo cd-initial-admin-secret -o
```

```
jsonpath=".data.password" | base64 --decode; echo
```

```
~/De/sd5976_aws_infrastructure | main ➔ kubectl -n argo cd get secret argo cd-initial-admin-secret -o jsonpath=".data.password" | base64 --decode; echo  
GOU50PhpbSyKFR
```

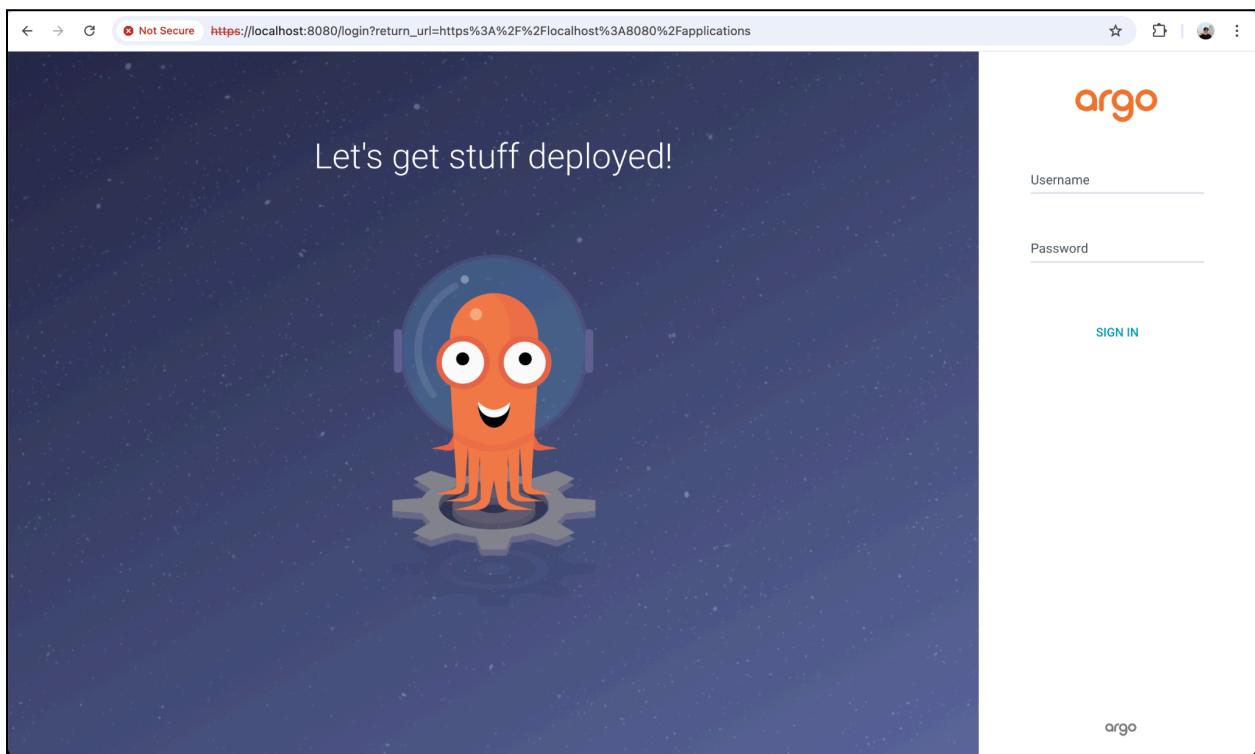
```
~/Desktop/sd5976_aws_infrastructure | main ➔
```

Forward the ArgoCD service port from EKS to your local machine:

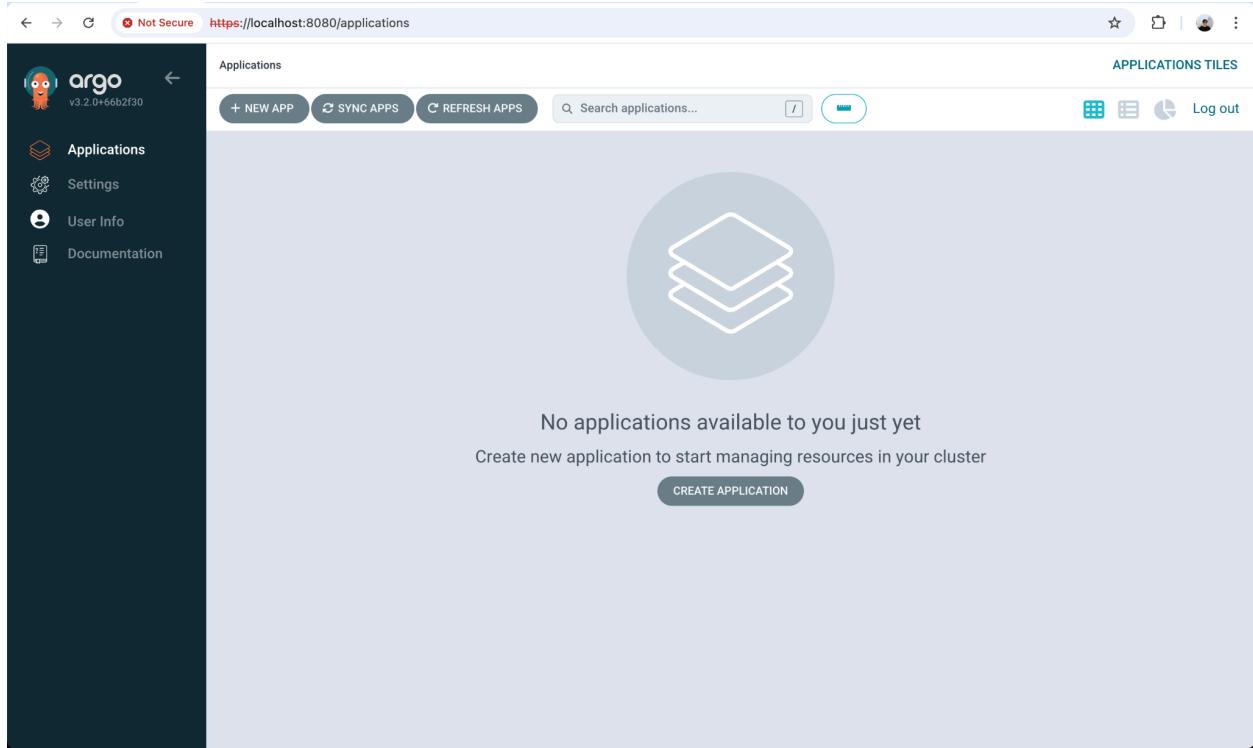
```
kubectl port-forward svc/argo cd-server -n argo cd 8080:443
```

```
~/De/sd5976_aws_infrastructure | main ➔ kubectl port-forward svc/argo cd-server -n argo cd 8080:443  
Forwarding from 127.0.0.1:8080 → 8080  
Forwarding from [::1]:8080 → 8080
```

Then open your browser and go to <https://localhost:8080>



Login to ArgoCD with default username **admin** and the decode password.



Once you can access the ArgoCD web interface, log in through the CLI:

```
argocd login localhost:8080 --insecure
```

```
~/Desktop/sd5976_aws_infrastructure | main ➔ argocd login localhost:8080 --insecure
Username: admin
Password:
'admin:login' logged in successfully
Context 'localhost:8080' updated
```

```
~/Desktop/sd5976_aws_infrastructure | main ➔
```

Go to **sd5976_devops_cicd_deployment/argocd-deployment**, run ecr-secret script to create ECR in EKS. This help ArgoCD can pull images from ECR inside EKS.

```
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main ➔ bash ecr-secret.sh -l app.kubernetes.io/name=argocd-image-updater -f
Creating ECR pull secret...
namespace/sd5976-msa configured
secret/ecr-secret created
ECR secret 'ecr-secret' created in namespace 'sd5976-msa'

Note: ECR tokens expire after 12 hours.
For production, set up a CronJob to refresh this secret.
```

Connect your deployment repository to ArgoCD so it can monitor and deploy automatically:

```
argocd repo add https://github.com/tri-dominh/sd5976_devops_cicd_deployment  
--username <your_username> --password <your_password>
```

```
~/Desktop/sd5976_devops_cicd_deployment/argocd-deployment | main ➔ argocd repo add https://github.com/tri-dominh/sd5976_devops_cicd_deployment --username tri-dominh  
--password ghp_...  
Repository 'https://github.com/tri-dominh/sd5976_devops_cicd_deployment' added  
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main ➔
```

Use **application.yaml** file to create **sd5976_msa** app in **argocd**:

```
kubectl apply -f application.yaml
```

```
argocd app get sd5976-msa
```

```
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main ➔ kubectl apply -f application.yaml  
Warning: metadata.finalizers: "resources-finalizer.argocd.argoproj.io": prefer a domain-qualified finalizer name to avoid accidental conflicts with other finalizer writers  
application.argoproj.io/sd5976-msa created  
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main ➔ argocd app get sd5976-msa  
Name: argocd/sd5976-msa  
Project: default  
Server: https://kubernetes.default.svc  
Namespace: sd5976-msa  
URL: https://localhost:8080/applications/sd5976-msa  
Source:  
- Repo: https://github.com/tri-dominh/sd5976_devops_cicd_deployment.git  
  Target: main  
  Path: argocd-deployment  
SyncWindow: Sync Allowed  
Sync Policy: Automated (Prune)  
Sync Status: Synced to main (7fb8472)  
Health Status: Healthy  


| GROUP       | KIND        | NAMESPACE  | NAME                        | STATUS  | HEALTH  | HOOK | MESSAGE                                         |
|-------------|-------------|------------|-----------------------------|---------|---------|------|-------------------------------------------------|
|             | Namespace   | sd5976-msa | sd5976-msa                  | Running | Synced  |      | namespace/sd5976-msa configured                 |
|             | Secret      | argocd     | ecr-credentials             | Synced  |         |      | secret/ecr-credentials configured               |
|             | Secret      | argocd     | git-creds                   | Synced  |         |      | secret/git-creds configured                     |
|             | ConfigMap   | argocd     | argocd-image-updater-config | Synced  |         |      | configmap/argocd-image-updater-config unchanged |
|             | Service     | sd5976-msa | backend                     | Synced  |         |      | service/backend unchanged                       |
|             | Service     | sd5976-msa | mongodb                     | Synced  | Healthy |      | service/mongodb unchanged                       |
|             | Service     | sd5976-msa | frontend                    | Synced  | Healthy |      | service/frontend unchanged                      |
| apps        | Deployment  | sd5976-msa | backend                     | Synced  | Healthy |      | deployment.apps/backend unchanged               |
| apps        | Deployment  | sd5976-msa | mongodb                     | Synced  | Healthy |      | deployment.apps/mongodb unchanged               |
| apps        | Deployment  | sd5976-msa | frontend                    | Synced  | Healthy |      | deployment.apps/frontend unchanged              |
| argoproj.io | Application | argocd     | sd5976-msa                  | Synced  |         |      | application.argoproj.io/sd5976-msa configured   |
|             | Namespace   |            | sd5976-msa                  | Synced  |         |      |                                                 |


```

Apply the **image-updater-config.yaml** file so that ArgoCD can automatically update images from ECR:

```
kubectl apply -f image-updater-config.yaml
```

```
~/De/sd5976_devops_cicd_deployment/argocd-deployment | main ➔ kubectl apply -f image-updater-config.yaml  
secret/ecr-credentials configured  
secret/git-creds configured  
configmap/argocd-image-updater-config configured
```

Once the deployment starts, you can monitor the pods running in the **sd5976-msa** namespace:

```
kubectl get pods -n sd5976-msa -w
```

~ / De / sd5976 _ devops _ cicd _ deployment / argocd - deployment main ! 1 ➔ kubectl get pods -n sd5976 - msa - w				
NAME	READY	STATUS	RESTARTS	AGE
backend - 8684865f6f - gsswz	1 / 1	Running	0	21m
backend - 8684865f6f - hrm92	1 / 1	Running	0	21m
frontend - 5687bcc76d - b7qvf	1 / 1	Running	0	21m
frontend - 5687bcc76d - c465t	1 / 1	Running	0	21m
mongodb - 78547887ff - t68k6	1 / 1	Running	0	21m

After all pods are running, get the external URL of your frontend service:

```
kubectl get svc frontend -n sd5976-msa
```

```
~/Desktop/sd5976_devops_cicd_deployment | main !] ➤ kubectl get svc frontend -n sd5976-msa ✓ | base Py | cicd-pipeline-cluster o
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP
frontend    LoadBalancer 172.20.97.235  a5a7251e8d40f4281b2652f5caf6a5b5-691894937.ap-southeast-1.elb.amazonaws.com
                                                       PORT(S)          AGE
                                                       80:32120/TCP   22m

~/Desktop/sd5976_devops_cicd_deployment | main !] ➤
```

The frontend application is now up and running.

