

School: CLOUDBOOSTA	Course: DevOps
Name: AKOSA SAMUEL ONYEJEKWE	Assignment-3_ Jenkins-2
Student No: CBALC0168	Issue Date: May 14, 2026

Tasks:

1. Configure Jenkins credentials management
2. Add Jenkins secret text credentials
3. Configure Jenkins database password secrets
4. Configure Jenkins username and password credentials
5. Configure MySQL database authentication credentials
6. Verify Jenkins global credentials storage
7. Update Jenkins stored credentials
8. Create Jenkins secure deployment pipeline
9. Configure secure Jenkins pipeline script
10. Configure Jenkins credential bindings
11. Execute secure deployment pipeline
12. Verify Jenkins credential masking
13. Verify secure pipeline execution
14. Install Jenkins Role-Based Authorization Strategy plugin
15. Configure Jenkins security authorization settings
16. Configure Jenkins role management
17. Create Jenkins user accounts
18. Assign Jenkins user roles
19. Verify Jenkins role-based access control
20. Verify Jenkins build history and security workflow
21. Verify Jenkins deployment execution logs
22. Terminate Jenkins EC2 server instance

Goal:

To configure secure Jenkins credentials management, implement Jenkins pipeline security practices, establish Role-Based Access Control (RBAC), manage secure deployment workflows, configure user authorization policies, and validate secure DevOps pipeline administration within a Jenkins cloud-native environment.

Result:

- Jenkins credentials management configured successfully
- Jenkins secret text credentials configured successfully
- Database password secrets configured successfully
- Jenkins username and password credentials configured successfully
- MySQL authentication credentials configured successfully
- Jenkins global credentials storage validated successfully
- Jenkins credentials updated successfully
- Jenkins secure deployment pipeline configured successfully
- Jenkins credential bindings configured successfully
- Secure deployment pipeline executed successfully
- Jenkins credential masking validated successfully
- Jenkins Role-Based Authorization Strategy plugin installed successfully
- Jenkins security authorization configured successfully
- Jenkins role management configured successfully
- Jenkins user accounts created successfully
- Jenkins user roles assigned successfully
- Jenkins role-based access control validated successfully
- Jenkins build history verified successfully
- Jenkins deployment execution logs validated successfully
- Jenkins EC2 server terminated successfully

Action:

Step 1: Verified Jenkins Global Credentials Store

- Opened: Jenkins → Manage Jenkins → Credentials → System → Global
- Verified existing credentials:
 - AWS credentials
 - GitHub access credentials
- Confirmed Jenkins global credentials configured successfully

Step 2: Added Jenkins Secret Text Credential

- Clicked: Add Credentials

- Selected credential type: Secret text
- Configured:
 - Scope: Global
 - Configured secret value successfully
- Configured credential ID: db-password
- Configured description: Database password for production MySQL
- Clicked: Create
- Verified secret text credential created successfully

Step 3: Added Username and Password Credential

- Clicked: Add Credentials
- Selected credential type: Username with password
- Configured:
 - Username: dbadmin
 - Configured password successfully
- Configured credential ID: db-credentials
- Configured description: MySQL database admin login
- Clicked: Create
- Verified username and password credential created successfully

Step 4: Verified Jenkins Credentials Configuration

- Opened: Jenkins Global Credentials
- Verified configured credentials:
 - AWS credentials
 - GitHub credentials
 - Database password credentials

- Database admin login credentials
- Confirmed credentials stored securely within Jenkins

Step 5: Updated Jenkins Stored Credentials

- Opened credential: db-password
- Selected: Update credential
- Modified stored secret successfully
- Saved updated credential configuration
- Verified credential update completed successfully

Step 6: Created Secure Deployment Pipeline Job

- Opened: Jenkins Dashboard → New Item
- Configured item name: secure-deploy
- Selected: Pipeline
- Clicked: OK
- Verified secure deployment pipeline created successfully

Step 7: Configured Secure Jenkins Pipeline Script

- Opened: secure-deploy → Configure
- Configured pipeline definition: Pipeline script
- Configured secure credential binding stages using: withCredentials
- Configured:
 - Secret text credential usage
 - Username and password credential usage
- Enabled: Use Groovy Sandbox
- Saved secure deployment pipeline successfully

Step 8: Executed Secure Deployment Pipeline

- Opened: secure-deploy
- Selected: Build Now
- Verified Jenkins pipeline execution started successfully
- Confirmed build added to Jenkins build queue successfully

Step 9: Verified Jenkins Credential Masking

- Opened: Console Output
- Verified:
 - Secret values masked automatically
 - Password values hidden securely
 - Credential bindings executed successfully
- Confirmed Jenkins credential masking functioning correctly

Step 10: Verified Secure Deployment Pipeline Execution

- Verified Jenkins pipeline stages:
 - Using Secret Text
 - Using Username/Password
- Confirmed:
 - Database connection simulation executed successfully
 - Login credential validation executed successfully
- Jenkins pipeline execution status: SUCCESS

Step 11: Verified Jenkins Dashboard Build Status

- Opened: Jenkins Dashboard

- Verified configured jobs:
 - continuous-deployment-job
 - continuous-integration-job
 - secure-deploy
- Confirmed successful build status displayed successfully

Step 12: Installed Role-Based Authorization Strategy Plugin

- Opened: Manage Jenkins → Plugins → Available Plugins
- Searched: role
- Selected plugin: Role-based Authorization Strategy
- Clicked: Install
- Verified plugin installation completed successfully

Step 13: Verified Plugin Download Progress

- Opened: Plugins → Download Progress
- Verified:
 - Internet connectivity successful
 - Plugin download successful
- Installation status: Success
- Confirmed plugin installed successfully

Step 14: Configured Jenkins Security Authorization

- Opened: Manage Jenkins → Security
- Configured:
 - Security Realm: Jenkins' own user database
- Configured authorization method: Role-Based Strategy

- Clicked: Save
- Verified Jenkins security authorization configured successfully

Step 15: Configured Jenkins Role Management

- Opened: Manage Jenkins → Manage and Assign Roles → Manage Roles
- Configured roles:
 - admin
 - Viewer
 - Deployer
- Configured role permissions for:
 - Credentials
 - Build
 - Job
 - Workspace
 - Agent access
- Saved Jenkins role configuration successfully

Step 16: Created Jenkins User Account

- Opened: Manage Jenkins → Jenkins' own user database → Create User
- Configured:
 - Username: kevinsamuel
 - Configured password successfully
- Configured:
 - Full name: Kevin Judefo
 - Configured email address successfully
- Clicked: Create User

- Verified Jenkins user created successfully

Step 17: Verified Jenkins User Database

- Opened: Jenkins' own user database
- Verified configured users:
 - kevinsamuel
 - phoenixsamuel
- Confirmed Jenkins users created successfully

Step 18: Assigned Jenkins User Roles

- Opened: Manage and Assign Roles → Assign Roles
- Assigned:
 - Admin role
 - Viewer role
 - Deployer role
 - Assigned roles to configured Jenkins users successfully
- Saved role assignment configuration

Step 19: Verified Jenkins Role-Based Access Control

- Verified:
- User permissions enforced successfully
- Role-based authorization functioning correctly
- Jenkins access management configured successfully

Step 20: Verified Jenkins Build History

- Opened: Build History of Jenkins
- Verified successful builds:

- secure-deploy
 - continuous-deployment-job
 - continuous-integration-job
- Verified stable pipeline execution history successfully

Step 21: Verified Deployment Execution Logs

- Opened: secure-deploy → Build #3
- Verified:
 - Build execution details
 - Build duration
 - Pipeline execution logs
 - Successful deployment workflow execution
- Confirmed deployment execution completed successfully

Step 22: Terminated Jenkins EC2 Server Instance

- Opened: AWS EC2 Console → Instances
- Selected instance: jenkins_server
- Selected: Instance State → Terminate Instance
- Verified instance termination initiated successfully
- Confirmed Jenkins EC2 server terminated successfully

Summary

- Jenkins credentials management configured successfully
- Secure Jenkins credential bindings validated successfully
- Secure deployment pipeline configured successfully
- Jenkins secret masking functionality validated successfully
- Jenkins Role-Based Authorization Strategy plugin installed successfully

- Jenkins RBAC security configuration validated successfully
- Jenkins user management and role assignment configured successfully
- Jenkins build history and deployment workflows verified successfully
- Jenkins cloud-native security and DevOps administration workflow validated successfully

Lab objective achieved:

Implemented Jenkins credentials management, secure deployment pipeline configuration, Role-Based Access Control (RBAC), Jenkins security authorization policies, user access management, secure credential masking, and cloud-native DevOps security workflow orchestration successfully within Jenkins CI/CD infrastructure.



Global

+ Add Credentials

Credentials that should be available everywhere.

	my-cba-aws-credential AKIAQESY6I3VY3LHXXD3	 
Login details to access my AWS Account		
	github-access samuelakosaonyejekwe/*****	 
github access		

 **Jenkins**

Manage Jenkins

Credentials

Global

Credentials that should be available everywhere

 my-cba-aws-credential AKIAQ3...
Login details to access my AWS Account

 github-access samuelakosajones...
github access

Add Credentials

REST API

Jenkins 2.555.1

Add Credentials

Select a type of credential

 **Username with password**
Commonly used for authentication to services like Git, APIs, or registries.

 **AWS Credentials**

 **GitHub App**

 **SSH Username with private key**

 **Secret file**

 **Secret text**

 **Certificate**
Upload a PKCS#12 or PEM encoded certificate and private key.

Next



Global

Credentials that should be available everywhere

- my-cba-aws-credential AKIAIS...
- github-access samuelkosaon...

Add Credentials

Add Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

db-password

Description ?

Database password for production MySQL

Create



Global

Credentials that should be available everywhere

my-cba-aws-credential AKIA3...
Login details to access my AWS Account

github-access samuelakosajones...
github access

db-password...
Database password for production...

+ Add Credentials

...

...

...

Add Credentials



Select a type of credential



Username with password

Commonly used for authentication to services like Git, APIs, or registries.



AWS Credentials



GitHub App



SSH Username with private key



Secret file



Secret text



Certificate

Upload a PKCS#12 or PEM encoded certificate and private key.

Next



Global

Credentials that should be available everywhere



my-cba-aws-credential

Login details to access my AWS from



github-access

github access



db-password

Database password for production M

+ Add Credentials



...



...



...

REST API

Jenkins 2.555.1

Add Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username

dbadmin

☐ Treat username as secret ?

Password

.....

ID ?

db-credentials

Description ?

MySQL database admin login

Create

Global

+ Add Credentials

Credentials that should be available everywhere.

 my-cba-aws-credential AKIAQESY6I3VY3LHXXK03 Login details to access my AWS Account	 
 github-access samuelakosonyejekwe/***** github access	 
 db-password Database password for production MySQL	 
 db-credentials dbodmny/***** MySQL database admin login	 

 **db-password**
Database password for production MySQL



 **db-credentials** dbadmin/*+*****
MySQL database admin login



Jenkins

Manage Jenkins

Credentials

Custom

Global

Global

Credentials that should be available everywhere

my-cba-aws-credential

AKIAQ3...

Login details to access my AWS from...

github-access

smuelakosawny...

github access

db-password

Database password for production M...

db-credentials

dbodmru*****

MySQL database admin login

+ Add Credentials

REST API

Jenkins 2.555.1

Update credential

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

Concealed

Change Password

ID ?

db-password

Description ?

Database password for production MySQL

Save

New Item

Enter an item name

secure-deploy

Select an item type



Pipeline

Build, test, and deploy using pipelines. Supports stages, parallel work, and running on multiple agents.



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.



Duplicate an existing item

Create a new item by copying the configuration of an existing one.

OK

Configure

General

Triggers

Pipeline

Advanced

General

Enabled 

Description

secure deploy

Plain text [Preview](#)

- ☐ Discard old builds ?
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☐ GitHub project
- ☐ Permission to Copy Artifact
- ☐ Pipeline speed/durability override ?
- ☐ Preserve stashes from completed builds ?
- ☐ This project is parameterized ?
- ☐ Throttle builds ?

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?
- ☐ Trigger builds remotely (e.g., from scripts) ?

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
18 ~      steps {
19 ~          withCredentials([usernamePassword(
20 ~              credentialsId: 'db-credentials',
21 ~              usernameVariable: 'DB_USER',
22 ~              passwordVariable: 'DB_PASS'
23 ~          )]) {
24 ~              sh '''
25 ~                  echo "Logging in as: $DB_USER"
26 ~                  echo "Password is: $DB_PASS"
27 ~              '''
28 ~          }
29 ~      }
30 ~  }
31 ~ }
32 ~ }
```

[try sample Pipeline...](#)☒ Use Groovy Sandbox ?

Advanced

Advanced ▾

Save

Apply

Status

secure-deploy

Edit description

Changes

secure deploy

Build Now

Permalinks

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

Builds >

...

today

✓ #1 12:45PM

▼

Status

Changes

Console Output

Edit Build Information

Delete build '1'

Timings

Pipeline Overview

Restart from Stage

Replay

Pipeline Steps

Workspaces

Console

Download

Copy

View as plain text

```
Started by user Akosa Samuel Onyejekwe
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/secure-deploy
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Using Secret Text)
[Pipeline] withCredentials
Masking supported pattern matches of $DB_PASS
[Pipeline] {
[Pipeline] sh
+ echo Connecting to database...
Connecting to database...
+ echo Password length: 12 characters
Password length: 12 characters
+ echo Password value: ****
Password value: ****
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Using Username/Password)
[Pipeline] withCredentials
Masking supported pattern matches of $DB_PASS
[Pipeline] {
[Pipeline] sh
+ echo Logging in as: dbadmin
Logging in as: dbadmin
+ echo Password is: ****
Password is: ****
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

**Jenkins**

/ secure-deploy ▾ / #1 ▾

```
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Using Username/Password)
[Pipeline] withCredentials
Masking supported pattern matches of $DB_PASS
[Pipeline] {
[Pipeline] sh
+ echo Logging in as: dbadmin
Logging in as: dbadmin
+ echo Password is: ****
Password is: ****
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- + New Item
- Build History
- Project Relationship
- Check File Fingerprint

Add description

All +

S	W	Name 1	Last Success	Last Failure	Last Duration	
		continuous-deployment-job	33 min #3 Started by phoenixsamuel	48 min #1	54 sec	
		continuous-integration-job	1 hr 1 min #2	N/A	14 sec	
		secure-deploy	2 min 30 sec #1	N/A	1.7 sec	



Icon: S M L

nan

Plugins

 role

 Install

▼



 Updates

 Available plugins

 Installed plugins

 Advanced settings

Install	Name	Released	Health
☒	Role-based Authorization Strategy 848.va_a_ea_673cf0b_c Security Authentication and User Management Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.	5 mo 7 days ago	97

Plugins

- 📄 Updates
- 📦 Available plugins
- 🔗 Installed plugins
- ⚙️ Advanced settings
- ☰ Download progress

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Role-based Authorization Strategy ✔ 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



Authentication

☐ Disable "Keep me signed in" ?

Security Realm

Jenkins' own user database ?

☐ Allow users to sign up ?

Authorization

Role-Based Strategy ?

Markup Formatter

Save Apply

-  Manage Roles
-  Assign Roles
-  Permission Templates
-  Role Strategy Macros

Role		Overall		Credentials					Agent					Job												
		Administer	Manage	Read	Create	Delete	ManagedDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete
 admin																										
 Viewer																										
 Deployer																										

Role to add

Deployer

Save

Apply





Create User

Username

kevinsamuel

Password

Confirm password

Full name

Kevin Jideofo

E-mail address

samuelakosaonyekwe@gmail.com

Create User



Users 2

+ Create User

These users can log into Jenkins. This is a sub-set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID ↕		Name		
	kevinsamuel	Kevin Jideofo		
	phoenixsamuel	Akosa Samuel Onyejekwe		

- Manage Roles
- Assign Roles
- Permission Templates
- Role Strategy Macros

Assign Roles

Global roles

User/Group	Deployer	Viewer	admin
Anonymous	☐	☐	☐
Authenticated Users	☐	☐	☐
Akosa Samuel Onyejekwe	☐	☐	☒
Kevin Jideofa	☒	☐	☐

Add User Add Group

Item roles

User/Group

Anonymous

Authenticated Users

Add User Add Group

Agent roles

User/Group

Anonymous

Authenticated Users

Add User Add Group

Save Apply

Build History

Project Relationship

Check File Fingerprint

Build Queue

No builds in the queue.

Build Executor Status

0/2

Build History of Jenkins

S	Build	Time Since	Status
✓	secure-deploy #3	31 sec	stable
✓	secure-deploy #2	2 min 18 sec	stable
✓	secure-deploy #1	1 hr 53 min	stable
✓	continuous-deployment-job #3 Started by phoenixsamuel	2 hr 24 min	stable
✓	continuous-deployment-job #2 Started by phoenixsamuel	2 hr 27 min	back to normal
✗	continuous-deployment-job #1	2 hr 39 min	broken since this build
✓	continuous-integration-job #2	2 hr 52 min	stable
✓	continuous-integration-job #1	2 hr 58 min	stable

Icon: S M L

0/100



📄 Status

🔗 Changes

📄 Console Output

👁 View Build Information

🕒 Timings

🔗 Pipeline Overview

🔄 Restart from Stage

🔄 Rebuild

☰ Pipeline Steps

⬅ Previous Build

✅ #3 (May 9, 2026, 2:38:22 PM)



Started by user [Kevin Jidefo](#)

Started 5 min 35 sec ago
Took [1.3 sec](#)



This run spent:

- 14 ms waiting;
- 1.3 sec build duration;
- 1.3 sec total from scheduled to completion.



No changes.

- + New Item
- Build History
- Project Relationship
- Check File Fingerprint

Add description

All +

S	W	Name 1	Last Success	Last Failure	Last Duration	
		continuous-deployment-job	2 hr 33 min #3 Started by phoenixsamuel	2 hr 48 min #1	54 sec	
		continuous-integration-job	3 hr 1 min #2	N/A	14 sec	
		secure-deploy	9 min 42 sec #3	N/A	1.3 sec	



Icon: S M L

EC2 > Instances

EC2

Dashboard

AWS Global View 

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Capacity Manager [New](#)

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Trust Stores

Auto Scaling

Auto Scaling Groups

Settings

Successfully initiated termination (deletion) of i-076382a5c61ef5e51

Instances (1/1) [Info](#)

Connect

Instance state

Actions

Launch instances

Saved filter sets

Choose filter set

< 1 >



☒	Name 	Instance ID	Instance state	Instance type	Status check	Alarm status
☒	jenkins_server	i-076382a5c61ef5e51	Terminated  	t3.medium	-	View alarms

i-076382a5c61ef5e51 (jenkins_server)

 CloudShell

 Feedback

 Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates.

[Privacy](#)

[Terms](#)

[Cookie preferences](#)