



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#),  
including cookie policy.

[Sign up](#)[Sign in](#)

# How to Integrate HashiCorp Vault with Jenkins to secure your Secrets



Nandita Sahu · [Follow](#)

8 min read · Jun 28, 2022



41



4



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



HashiCorp Vault with Jenkins

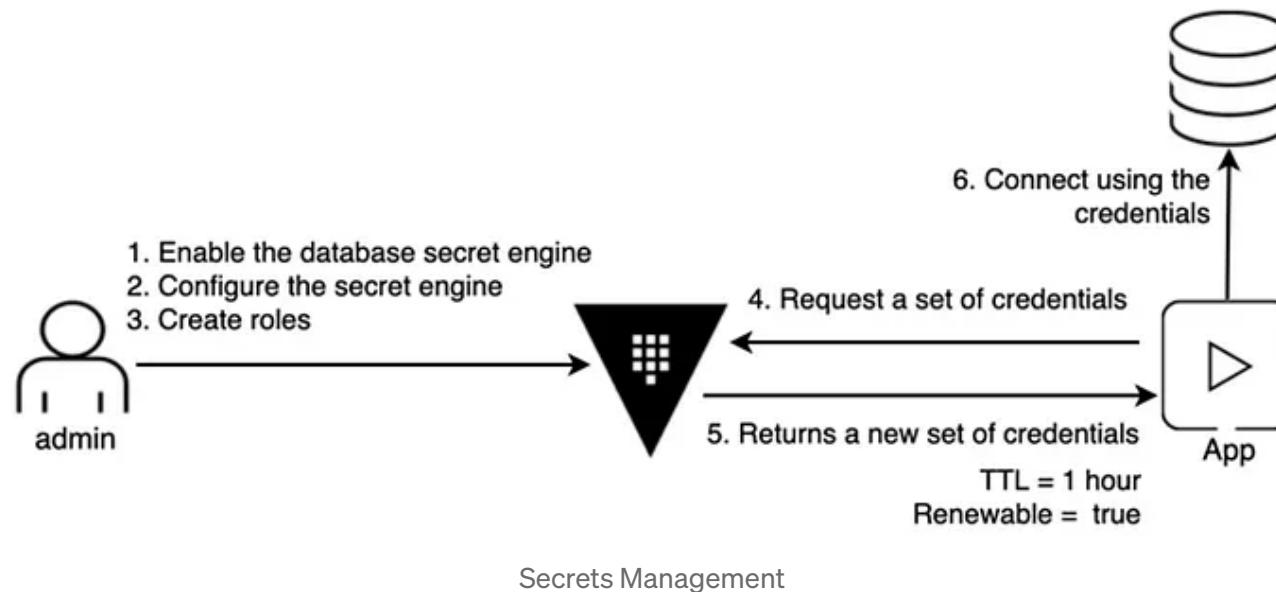
## What is Vault?

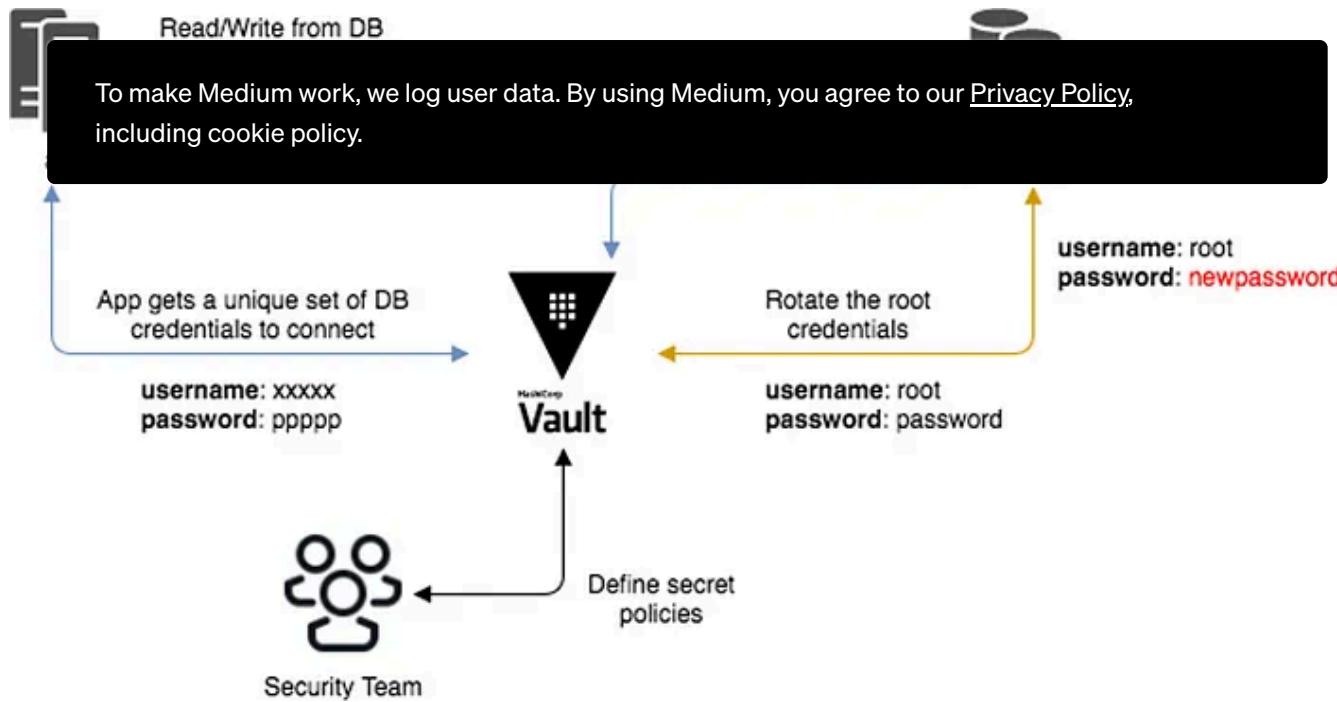
Vault is a tool that enables you to keep your secrets secured and encrypted. It can also do other things like encrypt data for your applications and generate temporary secrets and certificates for users and services. It is a cloud agnostic secret management system and safely store and manage sensitive data in hybrid cloud environment. It use to generate dynamic short-lived credentials or encrypt application data on fly.

There are different types of secrets : Username and Passwords, Certificates ,

En To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

## Vault Use Cases





**Secrets Management :** Centrally store, access and distribute secrets. Vault would be a best way to store sensitive environment variables, database credentials, API keys and many more .

**Encryption of Application Data :** Vault can be used to encrypt/decrypt data that is stored elsewhere. It allow applications to encrypt their data while still storing it in the primary data store.

**Identity Based Access :** Authenticate and access different clouds , systems and endpoints using trusted entities.

## Vault Server Modes

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#),  
including cookie policy.

Dev Mode for Development

Prod Mode for QA and Production

**Dev Mode :**

**Initialized and unsealed** — The server will be automatically initialized and unsealed. You don't need to use `vault operator unseal`. It is ready for use immediately.

**In-memory storage** — All data is stored (encrypted) in-memory. Vault server doesn't require any file permissions.

**Bound to local address without TLS** — The server is listening on `127.0.0.1:8200` (the default server address) *without* TLS.

**Automatically Authenticated** — The server stores your root access token so `vault` CLI access is ready to go. If you are accessing Vault via the API, you'll

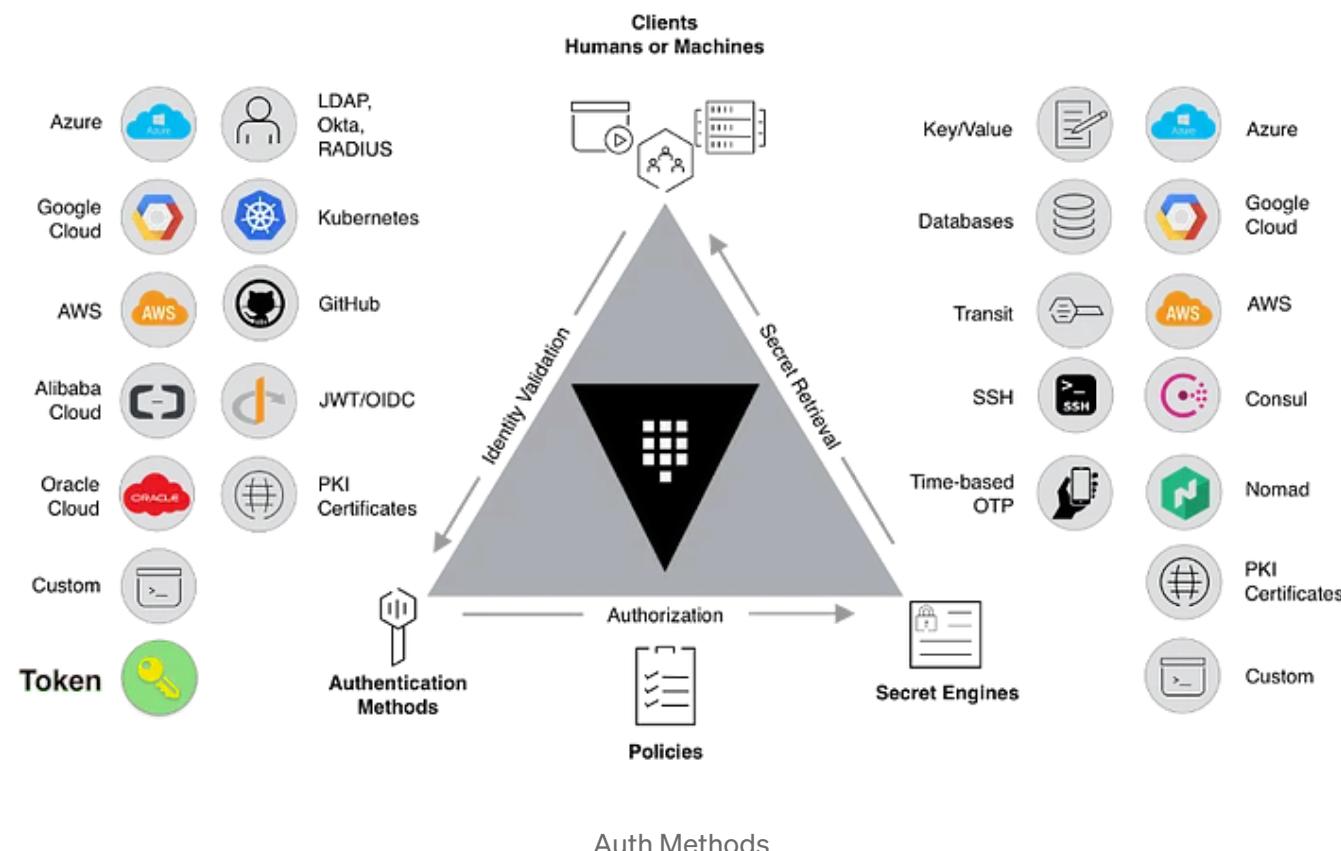
need to authenticate using the token printed out

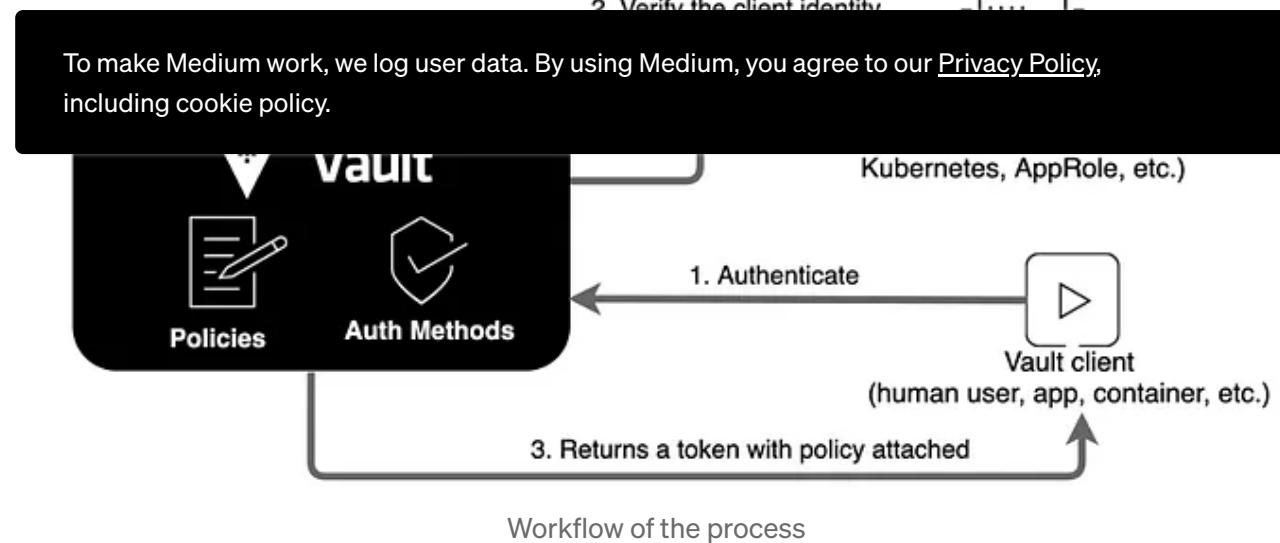
To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

[Sign In](#) | [Create Account](#) | [Log Out](#)

Vault is already unsealed, but if you want to experiment with seal/unseal, then only the single outputted key is required.

## Types of Auth Methods





Before a client can interact with Vault, it must authenticate with an auth method to acquire a token. This token has policies attached so that the behaviour of the client can be governed. Auth methods perform authentication to verify the user or machine-supplied information. Some of the supported auth methods are targeted towards users while others are targeted toward machines or apps.

## Why Vault instead of Jenkins Credentials ?

Problem with keeping our secret credentials in Jenkins is how pipeline handles credentials. Whenever a pipeline obtains a secret that is scoped, there are no limitations to it on how much and in what ways the pipeline can

use them. This poses a potential security threat considering the pipeline may be To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

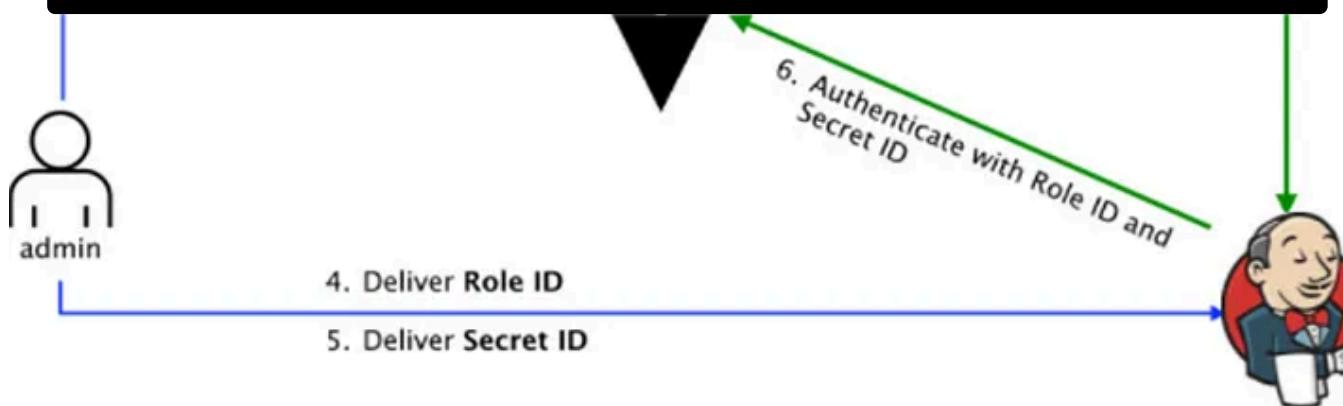
Another problem with Jenkins, its use of encryption methods. There is a static, single common master key and static single common hudson.util.Secret object. Both of these combine to encrypt our credentials. If access to our Jenkins host is compromised, we risk leaking these two keys and thus are credentials will be exposed.

The advantage of using Vault is

- It adds another layer of authentication over Jenkins.
- We can add multiple policies to access the same secret.
- Vault have rotation policy to any of the secrets you create.
- Centrally place to store, access and distribute secrets for organization.

### 1. Create policy and role for apps

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



Workflow Diagram of communication between Jenkins and Vault

### Installation of Vault (Ubuntu):

```
sudo apt update && sudo apt install gpg
```

```
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg >/dev/null
```

```
gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint
```

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
```

```
/etc/ansible/sources.list.d/hashicorp.list
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

```
sudo curl -L https://releases.hashicorp.com/vault/1.10.0/vault_1.10.0_amd64.deb -o /tmp/vault.deb
```

[Follow the steps for official Doc](#)

## Verifying the Installation

After installing Vault, verify the installation worked by opening a new terminal session and execute command “vault”, you should see help output similar to the following:

```
jenkins@devsecops:~$ vault
Usage: vault [command] [arguments]
To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy,  
including cookie policy.

write      Write data, configuration, and secrets
delete     Delete secrets and configuration
list       List data or secrets
login      Authenticate locally
agent      Start a Vault agent
server     Start a Vault server
status     Print seal and HA status
unwrap    Unwrap a wrapped secret

Other commands:
audit      Interact with audit devices
auth       Interact with auth methods
debug      Runs the debug command
kv         Interact with Vault's Key-Value storage
lease      Interact with leases
monitor   Stream log messages from a Vault server
namespace  Interact with namespaces
operator   Perform operator-specific tasks
path-help  Retrieve API help for paths
plugin    Interact with Vault plugins and catalog
policy    Interact with policies
print     Prints runtime configurations
secrets   Interact with secrets engines
ssh       Initiate an SSH session
token     Interact with tokens
version-history Prints the version history of the target Vault server
```

## Starting a Production Vault Server

Create the Vault configuration in the file “vault\_config.hcl”

```
listener "tcp" {
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

```
}
```

```
storage "file" {
```

```
    path = "./vault/data"
```

```
    node_id = "node1"
```

```
}
```

```
ui = true
```

```
api_addr = "http://localhost:8200"
```

```
cluster_addr = "http://127.0.0.1:8201"
```

```
disable_mlock = true
```

Configuration File

Within the configuration file, there are two primary configurations:

- **storage:** This is the physical backend that Vault uses for storage.

- ~~listener~~: One or more listeners determine how Vault listens for API requests.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

- `api_addr`: Specifies the address to advertise to route client requests.

- `cluster_addr`: Indicates the address and port to be used for communication between the Vault nodes in a cluster.

## Starting the Server

The `./vault/data` directory that file storage backend uses must exist.

```
$ mkdir -p ./vault/data
```

Set the `-config` flag to point to the proper path where you saved the configuration above.

If using DEV Mode on EC2 Instance run : `vault server -dev -dev-listen-address=<private ip of ec2>:8082`

If using PROD Mode on EC2 Instance run : `vault server -config=vault_config.hcl`

```
nansahu@vault-instance:/vault_demo$ vault server -config=vault_config.hcl
==> Vault server started! Log data will stream in below:

Listener 1: tcp (addr: "0.0.0.0:8200", cluster address: "0.0.0.0:8201", max_request_duration: "1m30s", max_request_size: "33554432", tls: "disabled")
  Log Level: info
    Mlock: supported: true, enabled: false
  Recovery Mode: false
    Storage: file
      Version: Vault v1.11.0, built 2022-06-17T15:48:44Z
      Version Sha: ea296ccf58507b25051bc0597379c467046eb2f1

==> Vault server started! Log data will stream in below:

2022-06-26T18:54:52.530Z [INFO] proxy environment: http_proxy="" https_proxy="" no_proxy=""
2022-06-26T18:54:52.530Z [INFO] core: Initializing version history cache for core
```

Launch a new terminal session, and set VAULT\_ADDR environment variable and check the status of the vault.

```
$ export VAULT_ADDR='http://127.0.0.1:8200'
```

```
$ vault status
```

```
|nansahu@vault-instance:~$ export VAULT_ADDR
[nan
To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy,
including cookie policy.
[nan
Key          Value
---          -----
Seal Type    shamir
Initialized   false
Sealed       true
Total Shares 0
Threshold    0
Unseal Progress 0/0
UnsealNonce  n/a
Version      1.11.0
Build Date   2022-06-17T15:48:44Z
Storage Type  file
HA Enabled   false
```

To initialize Vault use vault operator init. This is an unauthenticated request, but it only works on brand new Vaults without existing data:

```
|nansahu@vault-instance:/vault_demo$ ls
[vault]  vault_config.hcl
[nansahu@vault-instance:/vault_demo$ sudo vi vault_config.hcl
[nansahu@vault-instance:/vault_demo$ vault operator init -key-shares=1 -key-threshold=1
Unseal Key 1:

Initial Root Token:

Vault initialized with 1 key shares and a key threshold of 1. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 1 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated root key. Without at least 1 keys to
reconstruct the root key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.
```

Every initialized Vault server starts in the sealed state. From the command line, the vault operator unseal command is used to unseal the Vault. To unseal the Vault, a root token is required. A root token is a token that has the ability to perform all operations on the Vault. It is generated when the Vault is initialized. The process of using a root token to decrypt the data is known as unsealing the Vault.

```
$ export VAULT_TOKEN=<root token>
```

```
$ vault operator unseal
```

```
|nansahu@vault-instance:/vault_demo$ export VAULT_TOKEN=
|nansahu@vault-instance:/vault_demo$ vault operator unseal
Unseal Key (will be hidden):
Key          Value
---          -----
Seal Type    shamir
Initialized   true
Sealed       false
Total Shares 1
Threshold    1
Version      1.11.0
Build Date   2022-06-17T15:48:44Z
Storage Type file
Cluster Name vault-cluster-7a574740
Cluster ID   4e898b55-a892-e6d9-3d56-e031cc1f9c61
HA Enabled   false
```

When the value for `Sealed` changes to `false`, the Vault is unsealed.

Finally, authenticate as the initial root token (it was included in the output).

We use the vault command to log in. To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

```
$ vault login <root token>
```

```
nansahu@vault-instance:/vault_demo$ vault login
WARNING! The VAULT_TOKEN environment variable is set! The value of this
variable will take precedence; if this is unwanted please unset VAULT_TOKEN or
update its value accordingly.
```

```
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.
```

Key	Value
token	hvs.LtetcYQ8VavGi9tyQ2ATj9UH
token_accessor	aYzCdjidADBjflpB0iqrpgwA
token_duration	∞
token_renewable	false
token_policies	["root"]
identity_policies	[]
policies	["root"]

**Go to <IP Address>:8200 — → Shows the UI of the HashiCorp Vault Page**

## AppRole Authentication Method

An “AppRole” represents a set of Vault policies and login constraints that must be met to receive a token with those policies. An AppRole can be created for a particular machine, or even a particular user on that machine, or a service spread across machines.

## Enable the AppRole auth method:

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

```
$ vault auth enable approle
```

## Create a named role:

```
$ vault write auth/approle/role/jenkins-role token_num_uses=0  
secret_id_num_uses=0 policies="jenkins"
```

The screenshot shows the HashiCorp Vault UI interface. The top navigation bar includes 'Status', 'Secrets', 'Access', 'Policies', and 'Tools'. On the left, a sidebar titled 'ACCESS' has 'Auth Methods' selected, along with 'Multi-factor authentication', 'Entities', 'Groups', and 'Leases'. The main content area is titled 'approle' and contains the following configuration details:

- Configuration**:
  - Type: approle
  - Path: approle/
- Accessor**:
  - Local:  No
  - Seal wrap:  No
  - Default Lease TTL: 0
  - Max Lease TTL: 0
- Token Type**: (empty)

```
|nansahu@vault-instance:/vault_demo$ vault auth enable approle
Success! Enabled approle authentication back-end.

To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

Key          Value
-----
role_id      v1

|nansahu@vault-instance:/vault_demo$ vault write -f auth/approle/role/jenkins-role/secret-id
Key          Value
-----
secret_id    v1
secret_id_accessor 48e53cda-2576-8840-b20f-347d9eed4344
secret_id_ttl 0s
```

**token\_num\_uses:** If the token issued by your approle needs the ability to create child tokens, you will need to set token\_num\_uses to 0.

**secret\_id\_num\_uses=0:** Creating unlimited secrets

**policies:** Creating policy with the name of “Jenkins”

### Fetch the RoleID of the AppRole:

```
$ vault read auth/approle/role/jenkins-role/role-id
```

### Get a SecretID issued against the AppRole:

```
$ vault write -f auth/approle/role/jenkins-role/secret-id
```

## Install Vault Plugin & Integrate vault with Jenkins:

The screenshot shows the Jenkins Update Center interface. At the top, there is a black banner with white text: "To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy." Below this, the Jenkins logo is visible. The main area has tabs: "Updates" (selected), "Available", "Installed", and "Advanced". A search bar contains the text "hashicorp vault". The results list shows two items:

- HashiCorp Vault** 336.v182c0fbaeb7 (Released) - Build Wrappers. Jenkins plugin to populate environment variables from secrets stored in HashiCorp's Vault. A warning message is displayed: "Warning: This plugin version may not be safe to use. Please review the following security notices:" with a link to "Path traversal vulnerability allows reading arbitrary files". This item was released 6 months and 26 days ago.
- Hashicorp Vault Pipeline** 1.4 (Released) - Enables pulling of vault values as a pipeline step. This item was released 3 days 18 hours ago.

After installing the plugin, Navigate to Manage Credentials and add credentials and select credential type as **Vault AppRole Credentials** and fill out the **role ID**, **Secret ID**, **path** and **ID** as generic name to identify and click on Add.

The Role ID and Secret ID are ones which we got in the upper slide.

Vault App Role Credential

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

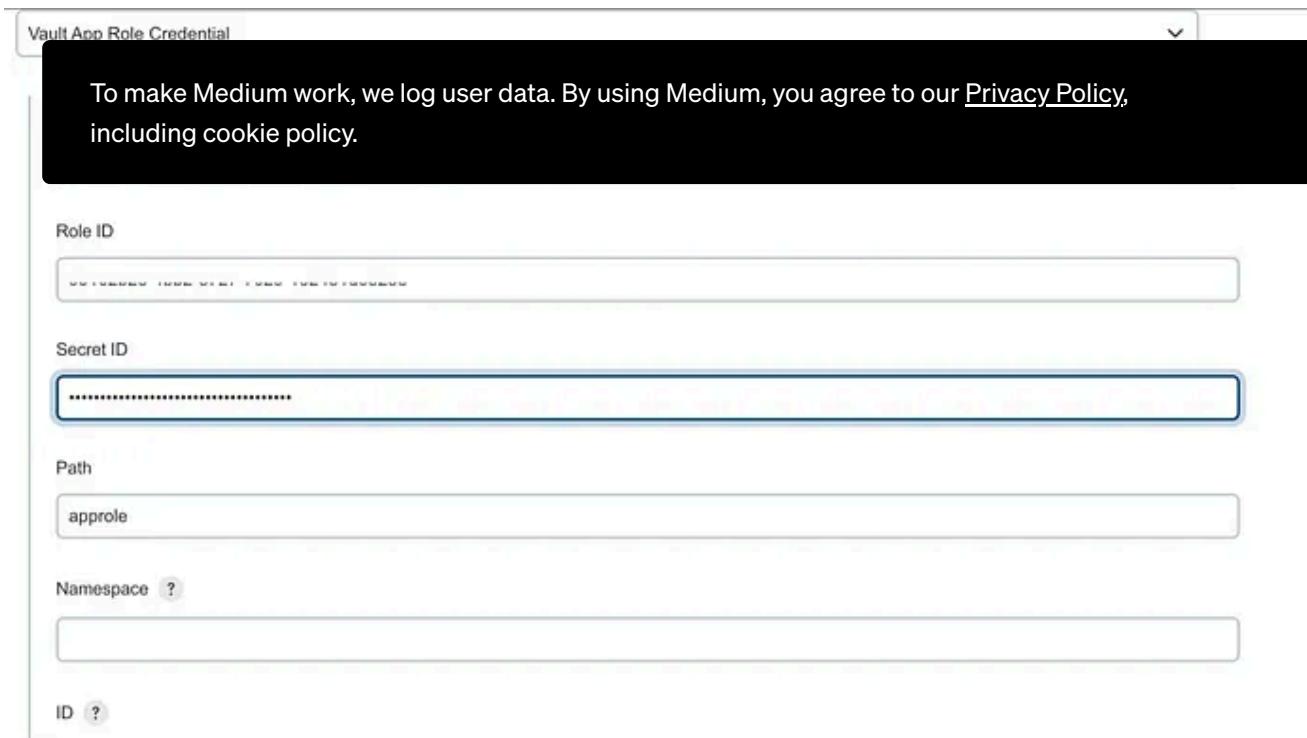
Role ID

Secret ID

Path

Namespace ?

ID ?



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

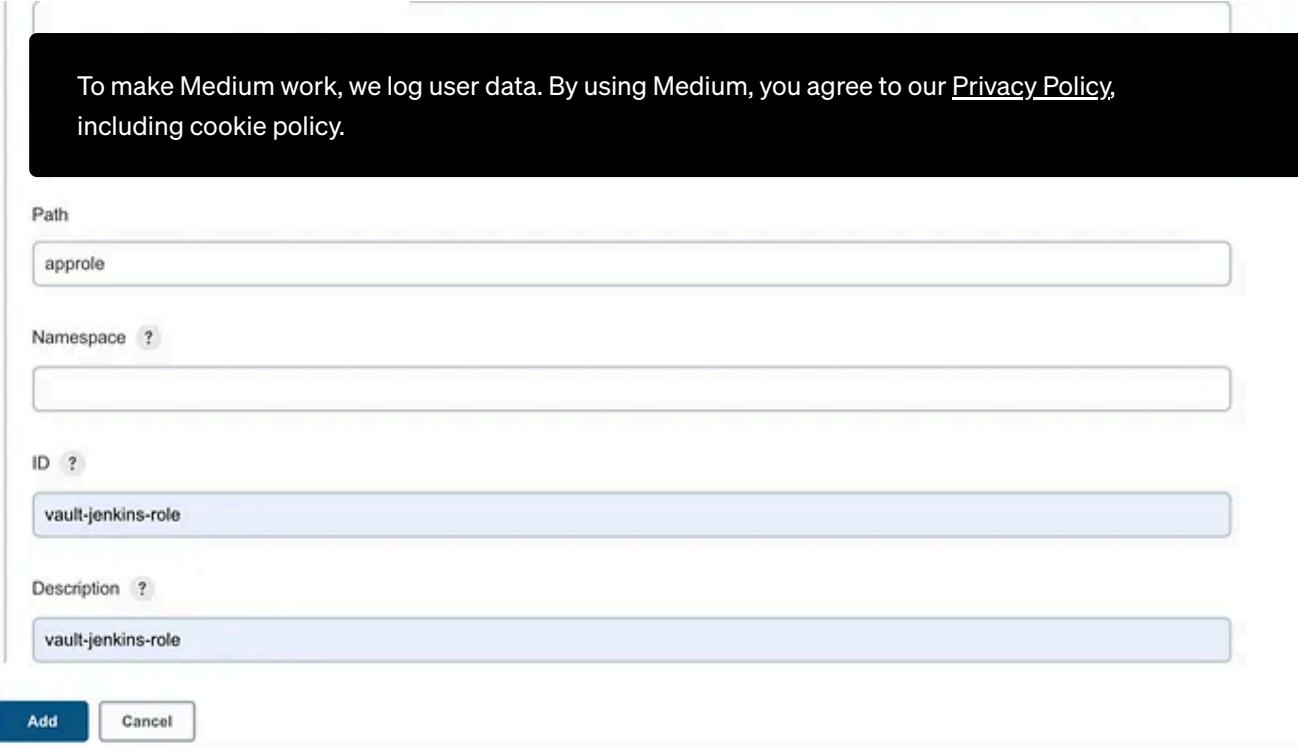
Path  
approle

Namespace ?

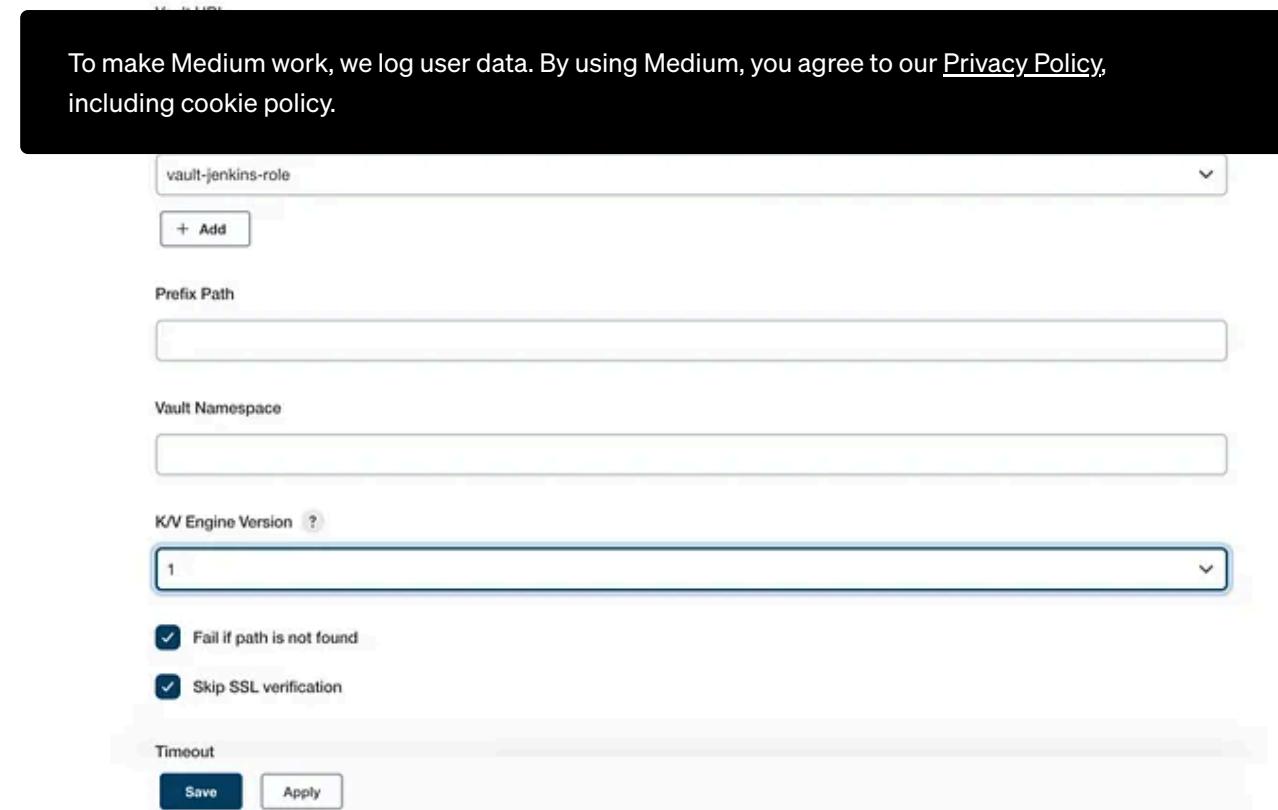
ID ?  
vault-jenkins-role

Description ?  
vault-jenkins-role

Add Cancel



## Vault plugin configure in Jenkins



Fill “Vault URL” (URL where Vault UI is accessible), “Vault Credential” (where we add the credentials mentioned in Jenkins for approle as vault-jenkins-role) .Click on advance settings to disable the ssl certification. (As we have not used SSL Certificates in demo).

## Create Secrets in Vault

Enable Secrets where path = “secrets” and it will use key value pair

```
$ vault secrets enable path-secrets kv
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

We

secret and value as jenkins123

```
$ vault write secrets/creds/my-secret-text secret=jenkins123
```

The screenshot shows the HashiCorp Vault interface. At the top, there's a breadcrumb navigation: < secrets < creds < my-secret-text. Below this, the path 'creds/my-secret-text' is displayed. A table titled 'Secret' lists one entry: 'secret' with a value of 'jenkins123'. The table includes columns for 'Key' and 'Value', and buttons for 'Delete', 'Copy', and 'Edit secret'.

Key	Value	Version created
secret	jenkins123	

We now create a policy to give permission to approle to retrieve secrets

```
$ vi jenkins-policy.hcl
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

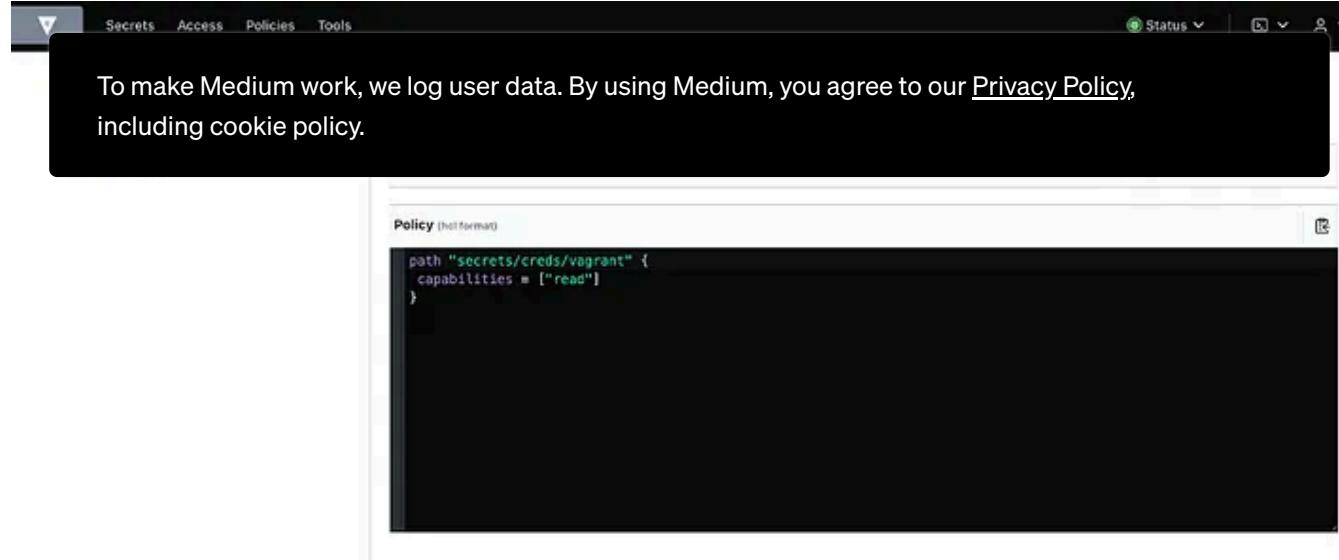
{}

### Policy for approle

This policy is giving read secrets permission to approle , those secrets which are stored in the path “secrets/creds/\*” (In vault)

```
$ vault policy write jenkins jenkins-policy.hcl
```

Create a policy named “jenkins” and use “jenkins-policy.hcl” as its content



Navigate to Manage Credentials and add credentials and select credential type as **Vault Secret Text Credentials** and fill “Path of the Secrets stored in Vault “ and ID .

The screenshot shows a web-based configuration interface for HashiCorp Vault. At the top, there's a navigation bar with links: Dashboard > Credentials > System > Global credentials (unrestricted) >. Below this is a black banner with white text that reads: "To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy." The main form area contains the following fields:

- Vault Key: A text input field containing "secrets/creds/my-secret-text".
- K/V Engine Version: A dropdown menu set to "1".
- ID: A text input field containing "vault-secret-text".
- Description: A text input field containing "vault-secret-text".

At the bottom left of the form, there's a message: "Successfully retrieved secret by key". On the right side, there's a blue button labeled "Test Vault Secrets retrieval".

## Create a Declarative Jenkins Pipeline

The screenshot shows a Jenkins pipeline configuration screen. At the top, there is a black banner with white text that reads: "To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy." Below this, the pipeline script is displayed in a code editor:

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 withCredentials([vaultString(credentialsId: 'vault-secret-text', variable: 'MYSECRET')]) {  
8                     sh '';  
9                     curl -X POST -H "Content-Type: application/json" -H "X-Custom-Header: $MYSECRET" http://34.122.6:  
10                '';  
11            }  
12        }  
13    }  
14}  
15}
```

Below the script, there is a checkbox labeled "Use Groovy Sandbox" which is checked.

Declarative Jenkins Pipeline

Here , we are using withCredentials function . We are using ‘credentialsId’ : Id of the secrets(my-secret-text) which we have declared in Jenkins Configure Credentials and ‘variable’ as ‘MYSECRET’ : which we will be using in jenkins pipeline referring to the ID of the secrets created in Jenkins.

The screenshot shows a 'Bindings' section for a 'Vault Secret Text Credential'. It includes fields for 'Variable' (set to 'MYSECRET') and 'Credentials' (set to 'secrets/creds/my-secret-text (vault-secret-text)'). A 'Generate Pipeline Script' button is present, which generates the following Jenkins declarative pipeline script:

```
withCredentials([vaultString(credentialsId: 'vault-secret-text', variable: 'MYSECRET')]) {
    // some block
}
```

## Output of the Jenkins Declarative Pipeline :

Started by user Nandita Sahu

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

```
[pipeline] stage
[Pipeline] { (Hello)
[Pipeline] withCredentials
Masking supported pattern matches of $MYSECRET
[Pipeline] {
[Pipeline] sh
+ curl -X POST -H Content-Type: application/json -H X-Custom-Header: ****
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total   Spent    Left  Speed
0       0     0       0       0       0       0 ---:---:--- ---:---:--- ---:---:--- 0
100  549  100  549     0       0  68625     0 ---:---:--- ---:---:--- ---:---:--- 134k
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
<title>Error 403 No valid crumb was included in the request</title>
</head>
<body><h2>HTTP ERROR 403 No valid crumb was included in the request</h2>
<table>
<tr><th>URI:</th><td>/</td></tr>
<tr><th>STATUS:</th><td>403</td></tr>
<tr><th>MESSAGE:</th><td>No valid crumb was included in the request</td></tr>
<tr><th>SERVLET:</th><td>Stapler</td></tr>
</table>
<hr/><a href="https://eclipse.org/jetty">Powered by Jetty:// 9.4.45.v20220203</a><hr/>

</body>
</html>
```

Output of Jenkins Job

If you like my article , Buy me a coffee :)

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



[Buy me a coffee](#)

For more references , go through the below videos and links

<https://gist.github.com/darinpope/4e46fad6d823ae249dec625bff3a2d82>

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

[Vault](#)[Jenkins Pipeline](#)[Jenkins](#)[Approle](#)[Secrets](#)

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



## Written by Nandita Sahu

1.4K Followers

Follow



I am quick learner and always love to explore new tools and technologies. You can buy me a coffee :) <https://www.buymeacoffee.com/NanditaSahu>

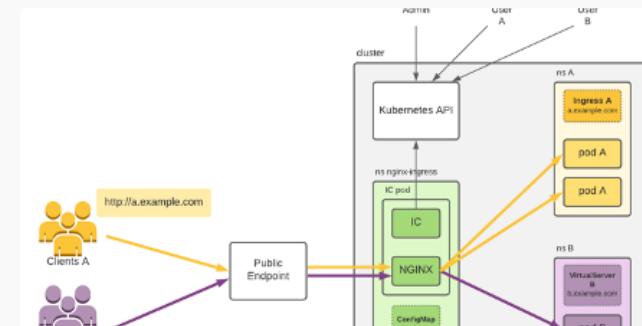
### More from Nandita Sahu



Nandita Sahu

## Terraformer—Generate Terraform Files from Existing Infrastructure

What is Terraformer?



Nandita Sahu

## “Securing Your EKS Cluster: A Step-by-Step Guide to...

Jul

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



Nandita Sahu

## How to define Dependencies between modules in Terragrunt

In this article , you will get an idea about how to define dependency blocks in Terragrunt...

Jan 4, 2023

20

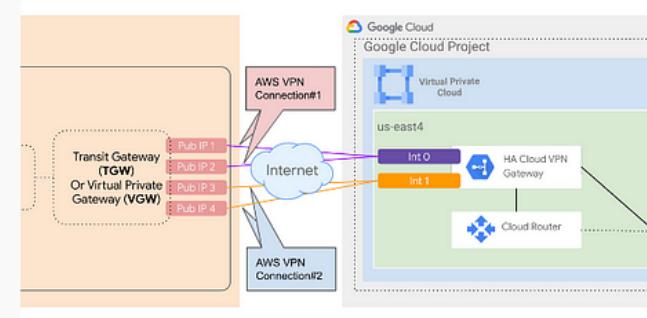
4



Oct 23, 2022

109

2



Nandita Sahu

## HA VPN connections between GCP and AWS

In this article, you will get a brief idea about how to create highly available VPN...

[See all from Nandita Sahu](#)

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

## Recommended from Medium



Kevin

### HashiCorp Vault—common use commands

a list of commonly used CLI commands for interacting with Vault



May 17



20



Oct 5



Khalid isyaku

### CICD Pipeline with Jenkins, Sonar and Docker

Project Introduction:

## Lists

**Staff Picks****Stories to Help You Level-Up**

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

**Self-Improvement 101**

20 stories · 2912 saves

**Productivity 101**

20 stories · 2467 saves



Bernardin Houessou

## DevSecOps dotnet pipeline : (Bitbucket, Jenkins, SonarQube,...)

DevSecOps : .Net , Jenkins, Sonarqube, Checkmarx, Owasp, Docker, K8s, Trivy

⭐ Apr 26

手掌 4



Cumhur Akkaya

## DevOps and Cloud Resources 2024 (PDF Books)

I updated DevOps and Cloud resources. I added 51 new documents. In this article; 🔎 ...

⭐ Sep 17

手掌 101

讲话 1



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



 Harpreet Singh Kalsi 

## Monitoring and DevSecOps with COTS and Custom Software

Recently I have had conversations in my professional setting wherein the points were...

 May 31  18



 DevOpsDynamo

## Conquer the CKA Exam! 5 Realistic Kubernetes Scenarios Every...

 if you're not a Medium member, read this story for free, [here](#).

 Oct 8  14  1



[See more recommendations](#)