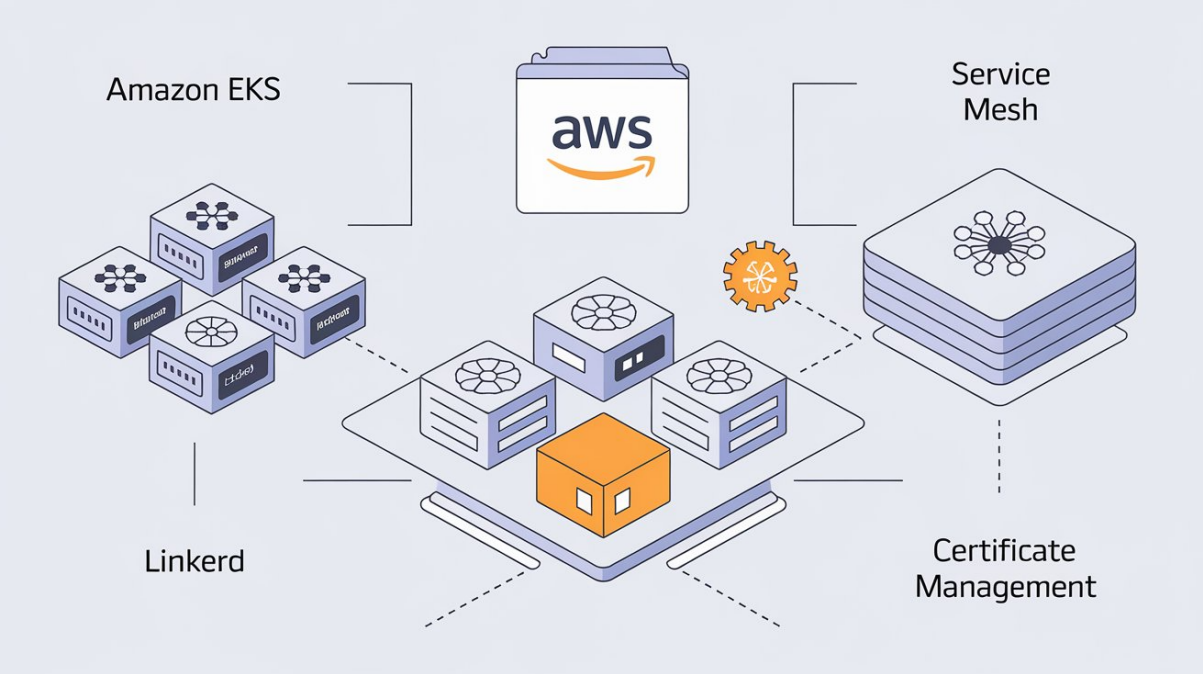
Effortless Certificate Management for Linkerd Service Mesh on AWS EKS Using HCP Vault Dedicated



By

Mahendran Selvakumar

https://devopstronaut.com/

https://www.linkedin.com/in/mahendran-selvakumar/

What is Linkerd Service Mesh?

Linkerd is a lightweight, open-source service mesh specifically designed for Kubernetes environments. It focuses on providing **security, observability, and reliability** for cloud-native applications by managing service-to-service communication within a cluster. Developed as a CNCF (Cloud Native Computing Foundation) project, Linkerd is known for its simplicity, efficiency, and ease of use.

Key Features of Linkerd

1. **Security:**
    - **mTLS (Mutual TLS):** Encrypts traffic between services automatically, ensuring secure communication without modifying application code.
    - Automatic certificate rotation and strong identity guarantees.
2. **Observability:**
    - Provides **golden metrics** (latency, success rate, and request volume) for services.
    - Tools for real-time inspection of service behavior, such as linkerd stat and linkerd tap.
    - Integration with observability tools like Prometheus, Grafana, and Jaeger for metrics and tracing.
3. **Reliability:**
    - Traffic policies for fine-grained control over communication between services.
    - Automatic retries and timeouts to handle transient failures.
    - Traffic splitting for blue-green or canary deployments.
4. **Lightweight:**
    - Minimal resource overhead compared to other service meshes like Istio.
    - Uses Rust-based proxies for efficiency and performance.
5. **Ease of Use:**
    - Simple installation and configuration process with a CLI (linkerd install).
    - Pre-configured defaults for Kubernetes clusters, making it beginner-friendly.

What is HCP Vault Dedicated?

HCP Vault Dedicated is a managed service by HashiCorp that provides a private, secure instance of Vault in the cloud. It's designed to help organizations manage secrets and dynamic credentials without the complexity of self-hosting.

Key Features:

- **Private Environment**: Ensures isolation and security
- **High Availability:** Guarantees uptime and reliability.

- **Simplified Operations**: HashiCorp handles maintenance and updates
- **Cloud Integration:** Works seamlessly with AWS, Azure, and other cloud platforms

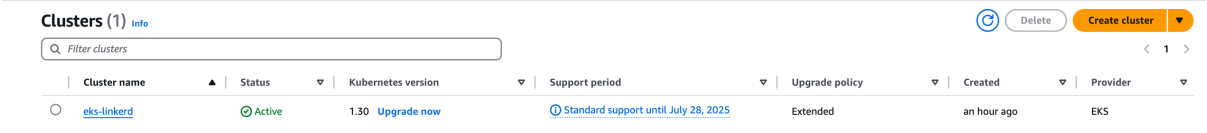
HCP Vault Dedicated simplifies secret management, improves security, and reduces operational overhead, making it a reliable solution for cloud-based environments

Step 1: Create the EKS Cluster Without Any Node Groups

Create an EKS cluster without a node group using the eksctl command **eksctl create cluster --name=eks-linkerd --region=eu-west-1 --without-nodegroup.** By default, eksctl creates a node group with m5.large instances, so we used the --without-nodegroup option to skip creating a default node group

```
mahendranselvakumar@Mahendrans-MBP ~ % eksctl create cluster --name=eks-linkerd --region=eu-west-1 --without-nodegroup
2024-12-31 14:04:56 [ℹ]  eksctl version 0.196.0
2024-12-31 14:04:56 [ℹ]  using region eu-west-1
2024-12-31 14:04:56 [ℹ]  setting availability zones to [eu-west-1a eu-west-1b eu-west-1c]
2024-12-31 14:04:56 [ℹ]  subnets for eu-west-1a - public:192.168.0.0/19 private:192.168.96.0/19
2024-12-31 14:04:56 [ℹ]  subnets for eu-west-1b - public:192.168.32.0/19 private:192.168.128.0/19
2024-12-31 14:04:56 [ℹ]  subnets for eu-west-1c - public:192.168.64.0/19 private:192.168.160.0/19
2024-12-31 14:04:56 [ℹ]  using Kubernetes version 1.30
2024-12-31 14:04:56 [ℹ]  creating EKS cluster "eks-linkerd" in "eu-west-1" region with
2024-12-31 14:04:56 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=eu-west-1 --cluster=eks-linkerd'
2024-12-31 14:04:56 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "eks-linkerd" in "eu-west-1"
2024-12-31 14:04:56 [ℹ]  CloudWatch logging will not be enabled for cluster "eks-linkerd" in "eu-west-1"
2024-12-31 14:04:56 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=eu-west-1 --cluster=eks-linkerd'
2024-12-31 14:04:56 [ℹ]  default addons coredns, vpc-cni, kube-proxy were not specified, will install them as EKS addons
2024-12-31 14:04:56 [ℹ]
2 sequential tasks: { create cluster control plane "eks-linkerd",
    2 sequential sub-tasks: {
        1 task: { create addons },
        wait for control plane to become ready,
    }
}
2024-12-31 14:04:56 [ℹ]  building cluster stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:04:57 [ℹ]  deploying stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:05:27 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:05:57 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:06:57 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:07:58 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:08:58 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:09:59 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:10:59 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:40:24 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-cluster"
2024-12-31 14:40:26 [ℹ]  creating addon
2024-12-31 14:40:26 [ℹ]  successfully created addon
2024-12-31 14:40:26 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested permissions; the recommended way to provide IAM permiss
ions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all recommended policies to the config file, under `addon.PodIdentityAssociations`, and run `eksctl update addon`
2024-12-31 14:40:26 [ℹ]  creating addon
2024-12-31 14:40:27 [ℹ]  successfully created addon
2024-12-31 14:40:27 [ℹ]  creating addon
2024-12-31 14:40:27 [ℹ]  successfully created addon
2024-12-31 14:43:29 [ℹ]  waiting for the control plane to become ready
2024-12-31 14:43:29 [✔]  saved kubeconfig as "/Users/mahendranselvakumar/.kube/config"
2024-12-31 14:43:29 [ℹ]  no tasks
2024-12-31 14:43:29 [✔]  all EKS cluster resources for "eks-linkerd" have been created
2024-12-31 14:43:29 [ℹ]  kubectl command should work with "/Users/mahendranselvakumar/.kube/config", try 'kubectl get nodes'
2024-12-31 14:43:29 [✔]  EKS cluster "eks-linkerd" in "eu-west-1" region is ready
mahendranselvakumar@Mahendrans-MBP ~ %
```

Go to the EKS console to verify that the cluster was successfully created using eksctl



Step 2: Create a Managed Node Group

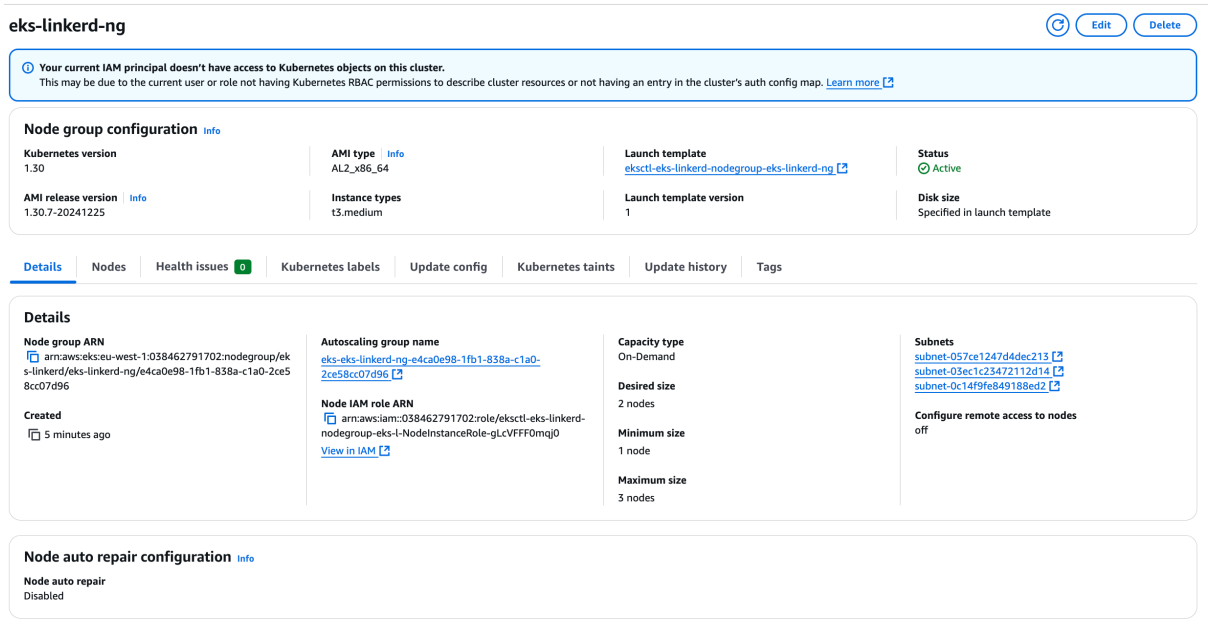
Add a node group using the following separate eksctl command **eksctl create nodegroup --name eks-linkerd-ng --cluster eks-linkerd --region eu-west-1 --nodes 2 --nodes-min 1 --nodes-max 3 --node-type t3.medium**

```
mahendranselvakumar@Mahendrans-MBP ~ % eksctl create nodegroup --name eks-linkerd-ng --cluster eks-linkerd --region eu-west-1 --nodes 2 --nodes-min 1 --nodes-max 3 --node-type t3.medium
2024-12-31 14:47:45 [ℹ]  will use version 1.30 for new nodegroup(s) based on control plane version
2024-12-31 14:47:46 [ℹ]  nodegroup "eks-linkerd-ng" will use "" [AmazonLinux2/1.30]
2024-12-31 14:47:47 [ℹ]  1 nodegroup (eks-linkerd-ng) was included (based on the include/exclude rules)
2024-12-31 14:47:47 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "eks-linkerd"
2024-12-31 14:47:47 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "eks-linkerd-ng" } } }
}
2024-12-31 14:47:47 [ℹ]  checking cluster stack for missing resources
2024-12-31 14:47:47 [ℹ]  cluster stack has all required resources
2024-12-31 14:47:47 [ℹ]  building managed nodegroup stack "eksctl-eks-linkerd-nodegroup-eks-linkerd-ng"
2024-12-31 14:47:48 [ℹ]  deploying stack "eksctl-eks-linkerd-nodegroup-eks-linkerd-ng"
2024-12-31 14:47:48 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-nodegroup-eks-linkerd-ng"
2024-12-31 14:48:18 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-nodegroup-eks-linkerd-ng"
2024-12-31 14:48:48 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-nodegroup-eks-linkerd-ng"
2024-12-31 14:50:05 [ℹ]  waiting for CloudFormation stack "eksctl-eks-linkerd-nodegroup-eks-linkerd-ng"
2024-12-31 14:50:05 [ℹ]  no tasks
2024-12-31 14:50:05 [✔]  created 0 nodegroup(s) in cluster "eks-linkerd"
2024-12-31 14:50:05 [ℹ]  nodegroup "eks-linkerd-ng" has 2 node(s)
2024-12-31 14:50:05 [ℹ]  node "ip-192-168-17-94.eu-west-1.compute.internal" is ready
2024-12-31 14:50:05 [ℹ]  node "ip-192-168-81-86.eu-west-1.compute.internal" is ready
2024-12-31 14:50:05 [ℹ]  waiting for at least 1 node(s) to become ready in "eks-linkerd-ng"
2024-12-31 14:50:05 [ℹ]  nodegroup "eks-linkerd-ng" has 2 node(s)
2024-12-31 14:50:05 [ℹ]  node "ip-192-168-17-94.eu-west-1.compute.internal" is ready
2024-12-31 14:50:05 [ℹ]  node "ip-192-168-81-86.eu-west-1.compute.internal" is ready
2024-12-31 14:50:05 [✔]  created 1 managed nodegroup(s) in cluster "eks-linkerd"
2024-12-31 14:50:05 [ℹ]  checking security group configuration for all nodegroups
2024-12-31 14:50:05 [ℹ]  all nodegroups have up-to-date cloudformation templates
mahendranselvakumar@Mahendrans-MBP ~ %
```

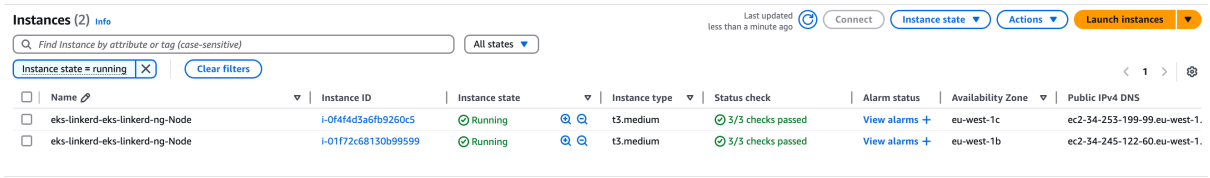
Explanation of the flags:

- --cluster: Specifies the name of the existing EKS cluster to which the node group will be added.
- --name: Names the node group for easy identification.
- -- region: Specifies the AWS region.
- --nodes: Sets the initial desired number of nodes (in this case, 2).
- -- nodes-min and — nodes-max: Define the minimum and maximum number of nodes for auto-scaling.
- --node-type: The EC2 instance type for the nodes (e.g., m5.large)

If you check in the console, you'll see that the node group has been created



In the instances section of the EC2 console, you can view the Instances created for the node group



Step 3: Configure Context for EKS Cluster

Set the Kubernetes context for the EKS cluster using the following command: **aws eks --region eu-west-1 update-kubeconfig --name eks-linkerd**

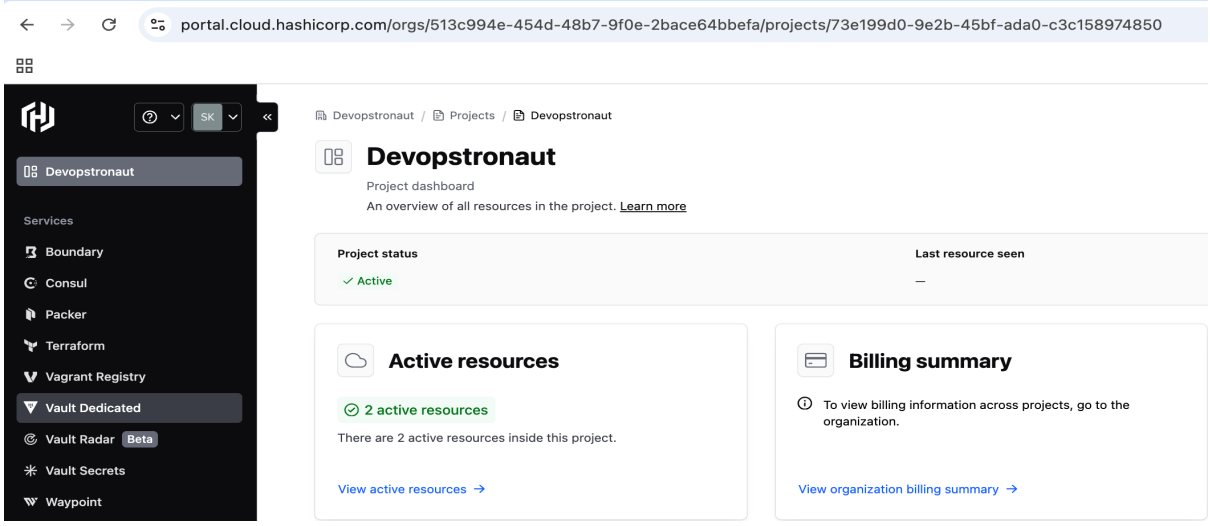
```
mahendranselvakumar@Mahendrans-MBP ~ % aws eks --region eu-west-1 update-kubeconfig --name eks-linkerd
Updated context arn:aws:eks:eu-west-1:038462791702:cluster/eks-linkerd in /Users/mahendranselvakumar/.kube/config
mahendranselvakumar@Mahendrans-MBP ~ %
```

Use **kubectl get ns** to view namespaces and **kubectl get nodes** to check the status of the nodes in the cluster, verifying that the setup is complete

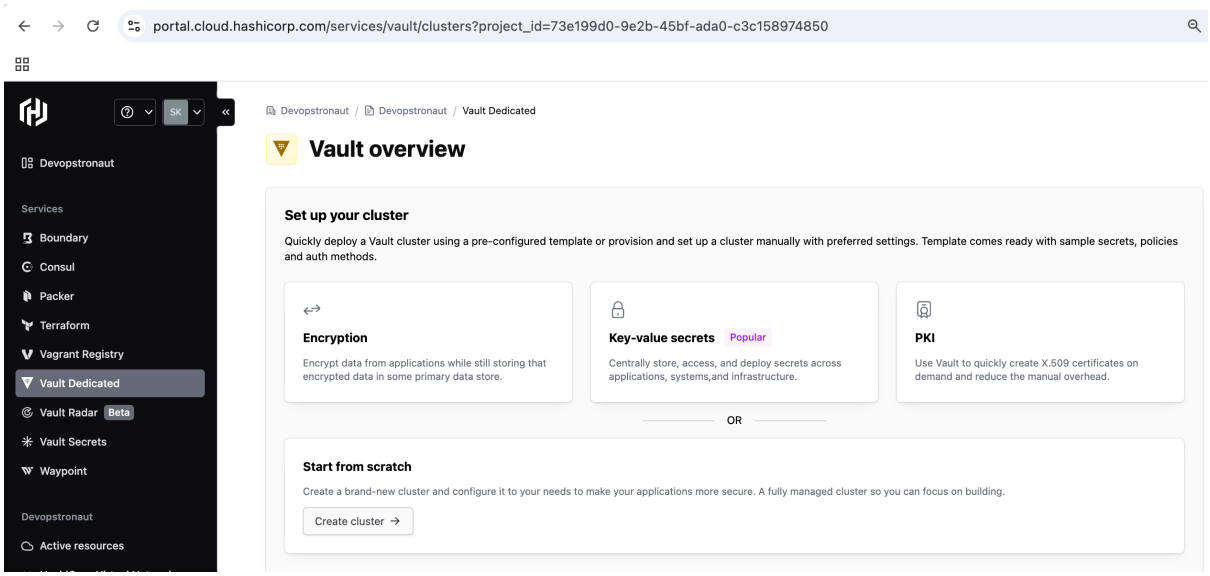
```
mahendranselvakumar@Mahendrans-MBP ~ % kubectl get ns
NAME              STATUS   AGE
default           Active   41m
kube-node-lease   Active   41m
kube-public       Active   41m
kube-system       Active   41m
mahendranselvakumar@Mahendrans-MBP ~ % kubectl get nodes
NAME                                           STATUS   ROLES    AGE    VERSION
ip-192-168-17-94.eu-west-1.compute.internal    Ready    <none>   2m6s   v1.30.7-eks-59bf375
ip-192-168-81-86.eu-west-1.compute.internal    Ready    <none>   2m4s   v1.30.7-eks-59bf375
mahendranselvakumar@Mahendrans-MBP ~ %
```

Step 4: Create the HCP Vault Dedicated

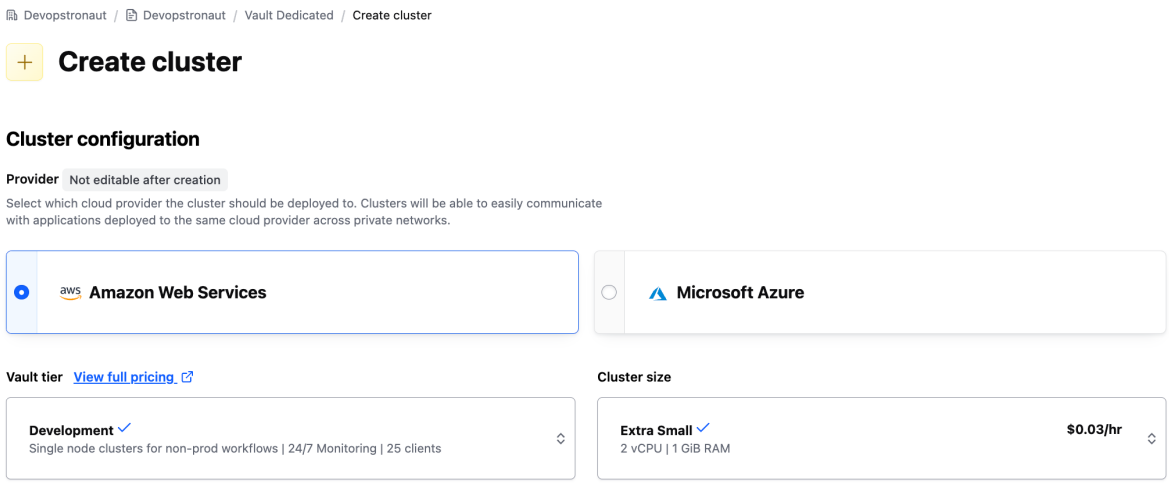
Click **Vault Dedicated**



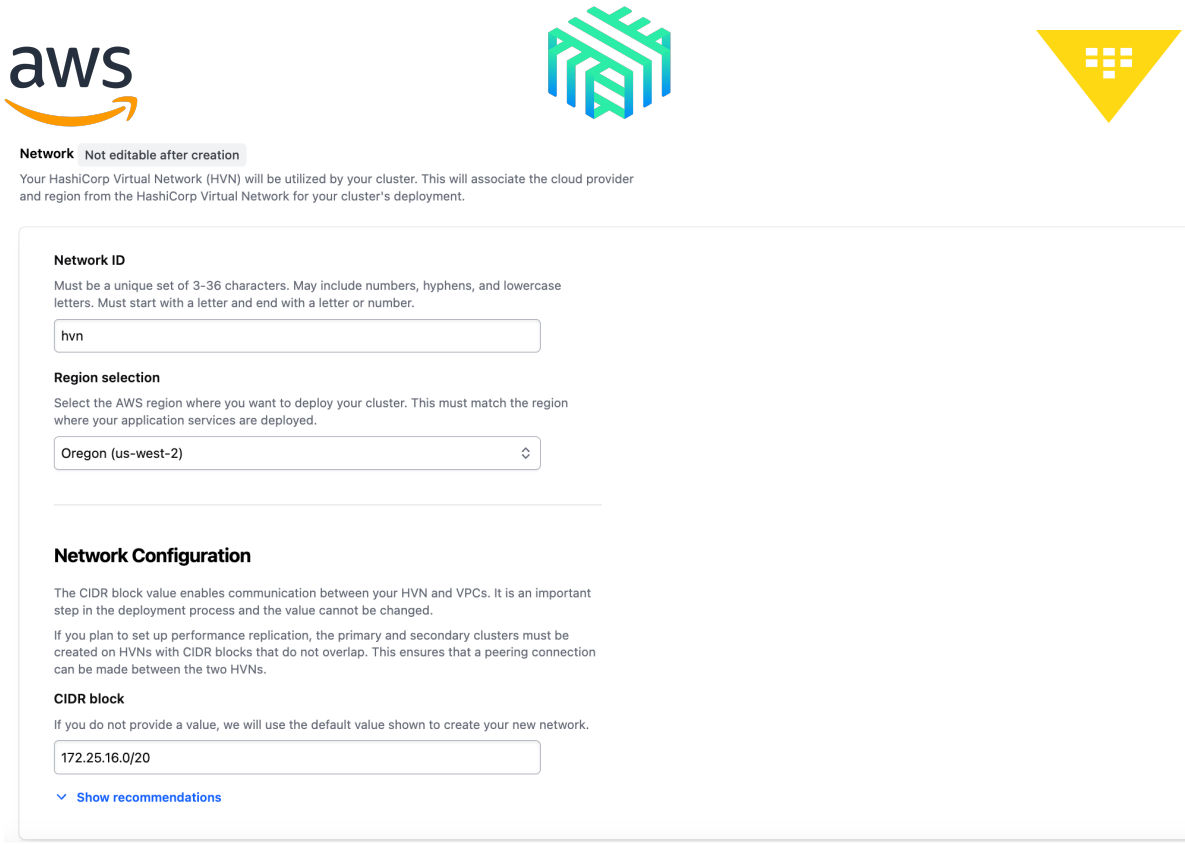
Click **Create Cluster** to create a cluster from scratch



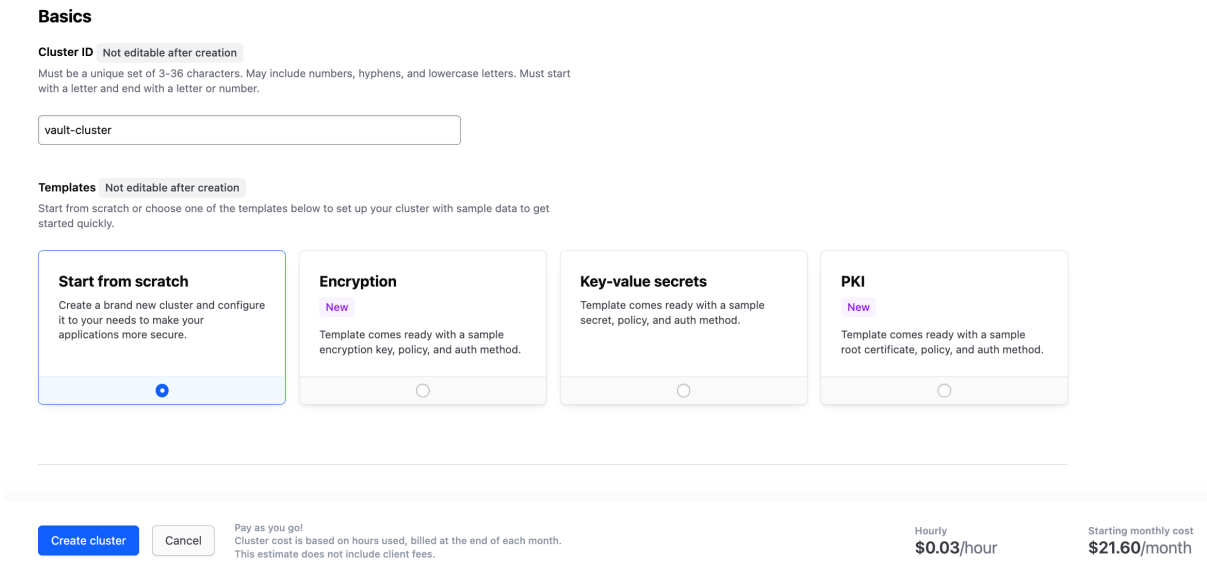
Choose **Amazon Web Services** as the provider and select the Vault tier based on your requirements



Enter the network ID and select the region

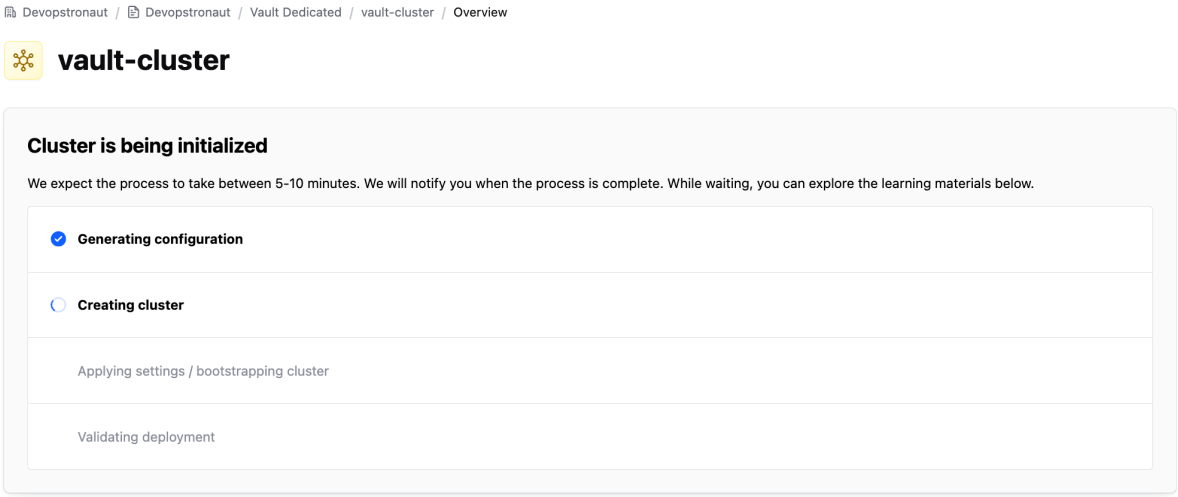


Enter the cluster ID, choose the template, and click **Create Cluster**

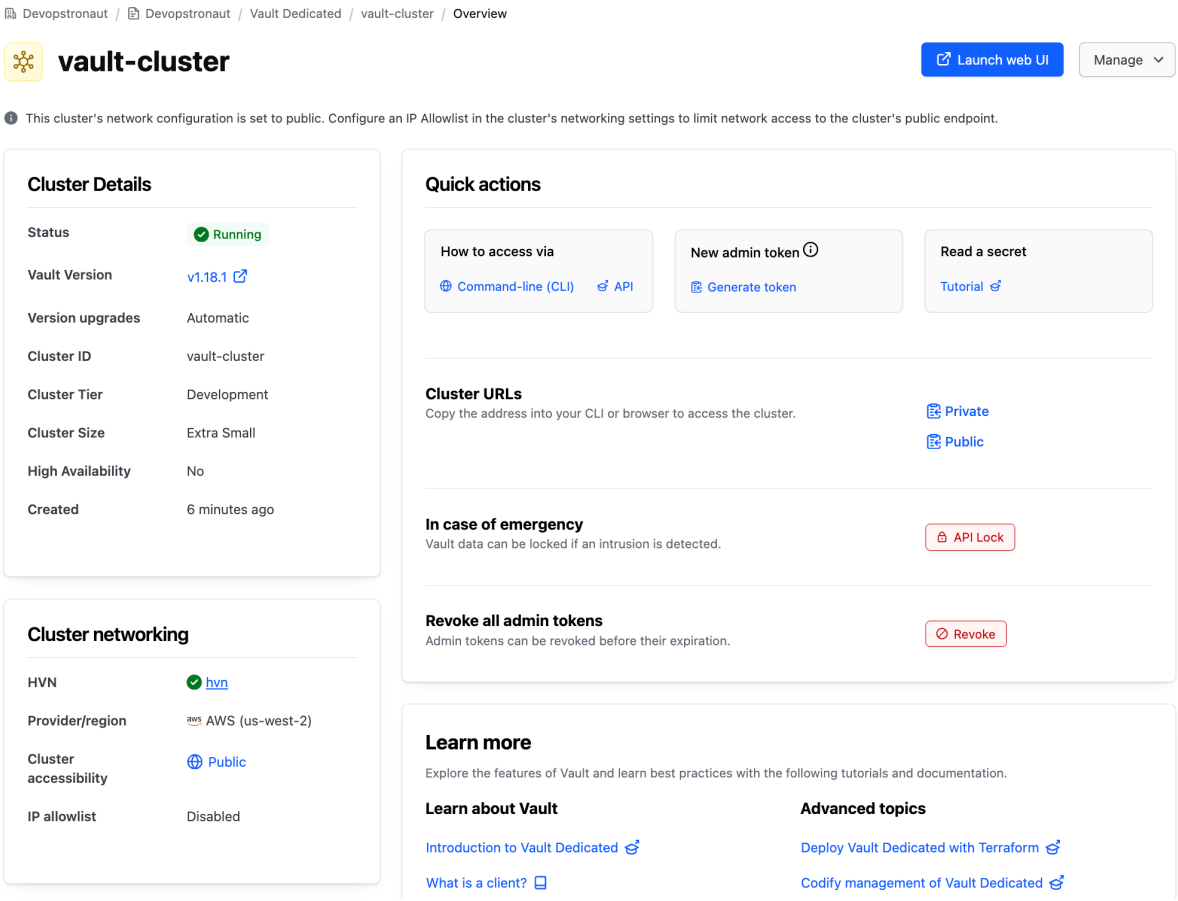


The Vault is now being created, and it may take some time to complete the process

https://www.linkedin.com/in/mahendran-selvakumar/

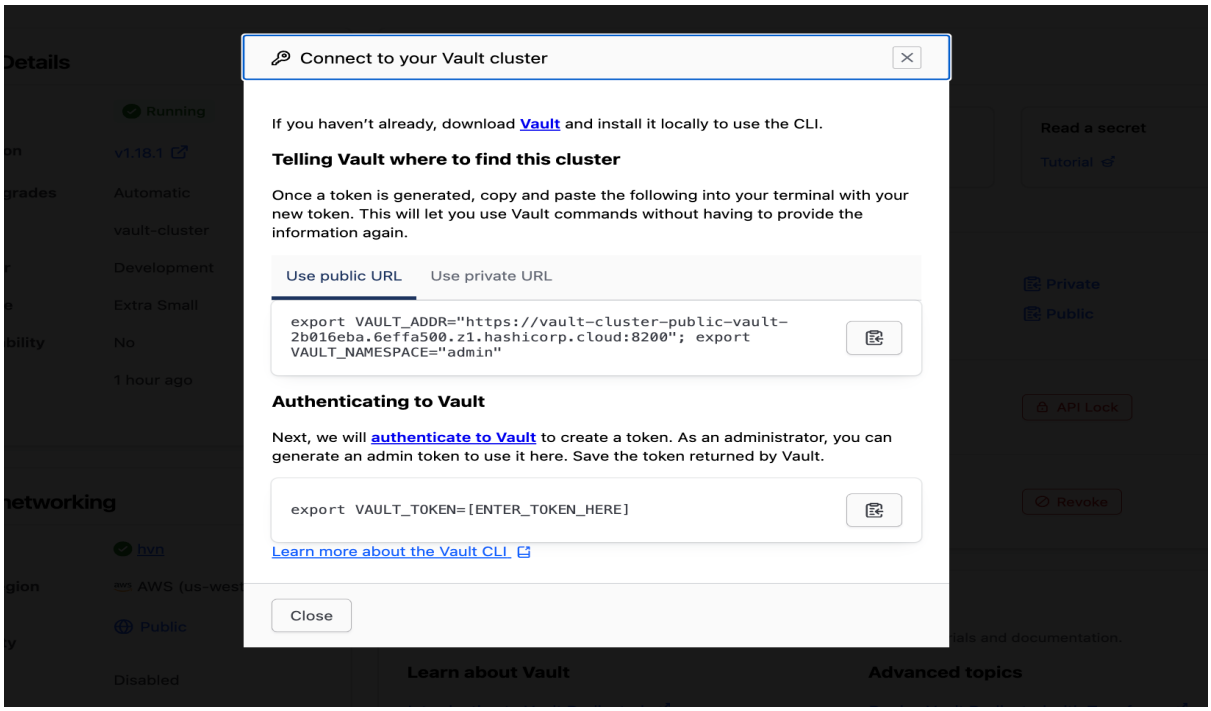


The Vault cluster has been successfully created, and you can view the cluster details, such as the version. Click **Command-line (CLI)** to view the Vault cluster and namespace details



https://www.linkedin.com/in/mahendran-selvakumar/

Copy the **Public URL** and **Namespace** to access the Vault cluster



Set Environment Variable

Run this command **export VAULT_ADDR=https://vault-cluster-public-vault-2b016eba.6effa500.z1.hashicorp.cloud:8200** to export Vault Address

```
mahendranselvakumar@Mahendrans-MBP ~ % export VAULT_ADDR="https://vault-cluster-public-vault-2b016eba.6effa500.z1.hashicorp.cloud:8200"
mahendranselvakumar@Mahendrans-MBP ~ %
```

Run this command **export VAULT_NAMESPACE="admin"** to export namespace

```
mahendranselvakumar@Mahendrans-MBP ~ % export VAULT_NAMESPACE="admin"
mahendranselvakumar@Mahendrans-MBP ~ %
```

Run this command **export VAULT_TOKEN=hvs.CAESIKYeXNkvRo7LQF2WUmdK1SshhcVPpqWZmXCPTJf_wxqIGicKImh2cy5PY1BsT1laVGhaTWp5WldUdW91WjVkWFcuMHZpUmMQ0yM** to set the root token

```
mahendranselvakumar@Mahendrans-MBP ~ % export VAULT_TOKEN=hvs.CAESIKYeXNkvRo7LQF2WUmdK1SshhcVPpqWZmXCPTJf_wxqIGicKImh2cy5PY1BsT1laVGhaTWp5WldUdW91WjVkWFcuMHZpUmMQ0yM
mahendranselvakumar@Mahendrans-MBP ~ %
```

Run the **vault status** command to view the Vault's status. You can see details such as the seal type, initialization and unsealing status (initialized and unsealed automatically), Vault version (Enterprise), and High Availability (HA) mode (active)

```
mahendranselvakumar@Mahendrans-MBP ~ % vault status
Key                     Value
---                     -----
Seal Type               awskms
Recovery Seal Type      shamir
Initialized             true
Sealed                  false
Total Recovery Shares   1
Threshold               1
Version                 1.18.1+ent
Build Date              2024-10-29T14:21:43Z
Storage Type            raft
Cluster Name            72047090-1af0-4f11-b6ce-7ccad412bef1
Cluster ID              4d322c8f-5d5b-8edf-352d-754e0d9cf158
HA Enabled              true
HA Cluster              https://172.25.16.49:8201
HA Mode                 active
Active Since            2024-12-31T13:59:40.802758651Z
Raft Committed Index    11435
Raft Applied Index      11435
Last WAL                4648
mahendranselvakumar@Mahendrans-MBP ~ %
```

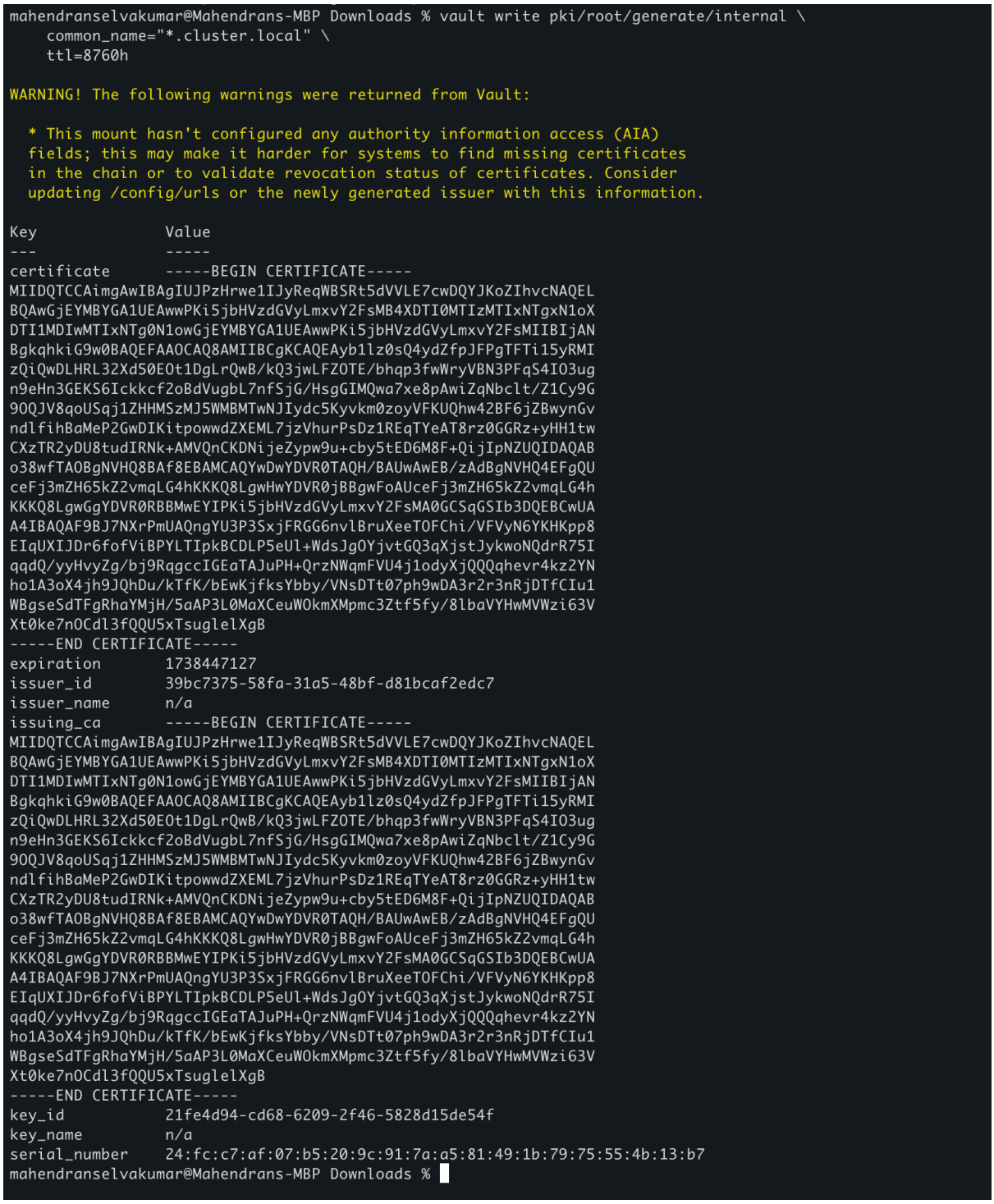
Step 5: Configure Vault for Certificate Issuance

Enable the PKI secrets engine for issuing TLS certificates using this command **Vault secrets enable pki**

```
mahendranselvakumar@Mahendrans-MBP ~ % vault secrets enable pki

Success! Enabled the pki secrets engine at: pki/
mahendranselvakumar@Mahendrans-MBP ~ %
```

Run the following command to generate a root PKI certificate with a common name of ***.cluster.local** and a validity of 8760 hours (1 year):

vault write pki/root/generate/internal common_name="*.cluster.local" ttl=8760h



Set up URLs for issuing certificates and distributing the Certificate Revocation List (CRL) using this command

vault write pki/config/urls issuing_certificates="$VAULT_ADDR/v1/pki/ca" crl_distribution_points="$VAULT_ADDR/v1/pki/crl"

```
mahendranselvakumar@Mahendrans-MBP Downloads % vault write pki/config/urls \
    issuing_certificates="$VAULT_ADDR/v1/pki/ca" \
    crl_distribution_points="$VAULT_ADDR/v1/pki/crl"
Key                        Value
---                        -----
crl_distribution_points    [https://vault-cluster-public-vault-b94a7a79.1f214b29.z1.hashicorp.cloud:8200/v1/pki/crl]
enable_templating          false
issuing_certificates       [https://vault-cluster-public-vault-b94a7a79.1f214b29.z1.hashicorp.cloud:8200/v1/pki/ca]
ocsp_servers               []
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Create a role for Linkerd using this command **vault write pki/roles/linkerd allowed_domains="identity.linkerd.cluster.local" allow_subdomains=true allow_any_name=true max_ttl="72h"**

```
mahendranselvakumar@Mahendrans-MBP Downloads % vault write pki/roles/linkerd \
    allowed_domains="identity.linkerd.cluster.local" \
    allow_subdomains=true \
    allow_any_name=true \
    max_ttl="72h"
Key                                   Value
---                                   -----
allow_any_name                        true
allow_bare_domains                    false
allow_glob_domains                    false
allow_ip_sans                         true
allow_localhost                       true
allow_subdomains                      true
allow_token_displayname               false
allow_wildcard_certificates           true
allowed_domains                       [identity.linkerd.cluster.local]
allowed_domains_template              false
allowed_other_sans                    []
allowed_serial_numbers                []
allowed_uri_sans                      []
allowed_uri_sans_template             false
allowed_user_ids                      []
basic_constraints_valid_for_non_ca    false
client_flag                           true
cn_validations                        [email hostname]
code_signing_flag                     false
country                               []
email_protection_flag                 false
enforce_hostnames                     true
ext_key_usage                         []
ext_key_usage_oids                    []
generate_lease                        false
issuer_ref                            default
key_bits                              2048
key_type                              rsa
key_usage                             [DigitalSignature KeyAgreement KeyEncipherment]
locality                              []
max_ttl                               72h
no_store                              false
no_store_metadata                     false
not_after                             n/a
not_before_duration                   30s
organization                          []
ou                                    []
policy_identifiers                    []
postal_code                           []
province                              []
require_cn                            true
server_flag                           true
signature_bits                        256
street_address                        []
ttl                                   0s
use_csr_common_name                   true
use_csr_sans                          true
use_pss                               false
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Request a certificate for Linkerd's identity service using this command **vault write pki/issue/linkerd common_name="identity.linkerd.cluster.local" ttl="72h"**

```
mahendranselvakumar@Mahendrans-MBP Downloads % vault write pki/issue/linkerd \
    common_name="identity.linkerd.cluster.local" \
    ttl="72h"
Key                 Value
---                 -----
ca_chain            [-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgIUJPzHrwe1IJyReqWBSRt5dVVLE7cwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMB4XDTI0MTIzMTIxNTgxN1oX
DTI1MDIwMTIxNTg0N1owGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyb1lz0sQ4ydZfpJFPgTFTi15yRMI
zQiQwDLHRL32Xd50EOt1DgLrQwB/kQ3jwLFZOTE/bhqp3fwWryVBN3PFqS4IO3ug
n9eHn3GEKS6Ickkcf2oBdVugbL7nfSjG/HsgGIMQwa7xe8pAwiZqNbclt/Z1Cy9G
9OQJV8qoUSqj1ZHHMSzMJ5WMBMTwNJIydc5Kyvkm0zoyVFKUQhw42BF6jZBwynGv
ndlfihBaMeP2GwDIKitpowwdZXEML7jzVhurPsDz1REqTYeAT8rz0GGRz+yHH1tw
CXzTR2yDU8tudIRNk+AMVQnCKDNijeZypw9u+cby5tED6M8F+QijIpNZUQIDAQAB
o38wfTAOBgNVHQ8BAf8EBAMCAQYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU
ceFj3mZH65kZ2vmqLG4hKKKQ8LgwHwYDVR0jBBgwFoAUceFj3mZH65kZ2vmqLG4h
KKKQ8LgwGgYDVR0RBBMwEYIPKi5jbHVzdGVyLmxvY2FsMA0GCSqGSIb3DQEBCwUA
A4IBAQAF9BJ7NXrPmUAQngYU3P3SxjFRGG6nvlBruXeeTOFChi/VFVyN6YKHKpp8
EIqUXIJDr6fofViBPYLTIpkBCDLP5eUl+WdsJgOYjvtGQ3qXjstJykwoNQdrR75I
qqdQ/yyHvyZg/bj9RqgccIGEaTAJuPH+QrzNWqmFVU4j1odyXjQQQqhevr4kz2YN
ho1A3oX4jh9JQhDu/kTfK/bEwKjfksYbby/VNsDTt07ph9wDA3r2r3nRjDTfCIu1
WBgseSdTFgRhaYMjH/5aAP3L0MaXCeuWOkmXMpmc3Ztf5fy/8lbaVYHwMVWzi63V
Xt0ke7nOCdl3fQQU5xTsuglelXgB
-----END CERTIFICATE-----]
certificate         -----BEGIN CERTIFICATE-----
MIIETzCCAzegAwIBAgIUaIP1HmGWCaHVuWRILmGP9v3z2OgwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMB4XDTI0MTIzMTIyMDYxOFoX
DTI1MDEwMzIyMDY0OFowKTEnMCUGA1UEAxMeaWRlbnRpdHkubGlua2VyZC5jbHVz
dGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv3C9LO6d
/iknwhxAvYTJytTCSnkzs0VWZHY2yhjW1/c3KlQ1m+CYEn3UPTbY9wegXEuK64r1
MySjC24cKEXuTR1+hu1yxmVs0DwwfXG/Y1ZrN+jjKaer7xPqphVjP5gRUO4aiu0X
lCbPY5DJPPsvkspjEzFwQQxLhhd6pKjtKstoi1AWVsJHo0EnxHq3gR5LXDIIE/1y
0r6q/lU4tshC3t8T5wQ+xgLmjgO6E3gqbZrW1rj1Oy2aQIEYytzMDWvEGJw6AuiM
ckLKuPSkDmYnFbwEZLehRGqf3K4VHUaYVvDN0uDwHAkGjqvagwnlB8OWKPUNajzX
F9avejMjj6SfqwIDAQABo4IBfDCCAXgwDgYDVR0PAQH/BAQDAgOoMB0GA1UdJQQW
MBQGCCsGAQUFBwMBBggrBgEFBQcDAjAdBgNVHQ4EFgQUNrvcPENu+H1WL2gb7tPM
jTG1PqQwHwYDVR0jBBgwFoAUceFj3mZH65kZ2vmqLG4hKKKQ8LgwcgYIKwYBBQUH
AQEEZjBkMGIGCCsGAQUFBzAChlZodHRwczovL3ZhdWx0LWNsdXN0ZXItcHVibGlj
LXZhdWx0LWI5NGE3YTc5LjFmMjE0YjI5LnoxLmhhc2hpY29ycC5jbG91ZDo4MjAw
L3YxL3BraS9jYTApBgNVHREEIjAggh5pZGVudGl0eS5saW5rZXJkLmNsdXN0ZXIu
bG9jYWwwaAYDVR0fBGEwXzBdoFugWYZXaHR0cHM6Ly92YXVsdC1jbHVzdGVyLXB1
YmxpYy12YXVsdC1iOTRhN2E3OS4xZjIxNGIyOS56MS5oYXNoaWNvcnAuY2xvdWQ6
ODIwMC92MS9wa2kvY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQCxnb2MXdsoQ4Wx96rc
c70ZAUEIbpnfXTF4p43UbTXp2oshu+ZsrMUd+PFmCf669+Y2TGvyen4qYtz0278t
v37urbYk17lqsD1zdg5eQPWeTTjMEzZk4l4/+0xtc3lexYwikTBYbJ6jcYfR7uox
6fOeCITYD1Rvr5EduOB8D52hqdd9CzFBZqyxyR92LB+40XW8R4ty9+Kz7A2ocqqP
KICDIihCIu3cDhHquYibYriOS1TyFMpjcpqCl2/Lqceurlc/EFlDMO4FO4LdtYsN
QV8LLoQpWU34pZUTxO/KLGxgsNWWe62pJ90DIcvA8Pdz/cn4POcZ4Y+B475me7Zh
db4J
-----END CERTIFICATE-----
expiration          1735942008
issuing_ca          -----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgIUJPzHrwe1IJyReqWBSRt5dVVLE7cwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMB4XDTI0MTIzMTIxNTgxN1oX
DTI1MDIwMTIxNTg0N1owGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyb1lz0sQ4ydZfpJFPgTFTi15yRMI
zQiQwDLHRL32Xd50EOt1DgLrQwB/kQ3jwLFZOTE/bhqp3fwWryVBN3PFqS4IO3ug
n9eHn3GEKS6Ickkcf2oBdVugbL7nfSjG/HsgGIMQwa7xe8pAwiZqNbclt/Z1Cy9G
```

https://www.linkedin.com/in/mahendran-selvakumar/

Generate the **linkerd-issuer.pem** and **linkerd-cert.pem** files using the certificate and key details provided in the above output

Combine certificates(cert.pem and ca_chain.pem) into a single PEM file(linkerd-issuer.pem)

```
mahendranselvakumar@Mahendrans-MBP Downloads % cat linkerd-issuer.pem
-----BEGIN CERTIFICATE-----
MIIETzCCAzegAwIBAgIUaIP1HmGWCaHVuWRILmGP9v3z2OgwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMB4XDTI0MTIzMTIyMDYxOFoX
DTI1MDEwMzIyMDY0OFowKTEnMCUGA1UEAxMeaWRlbnRpdHkubGlua2VyZC5jbHVz
dGVyLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv3C9LO6d
/iknwhxAvYTJytTCSnkzs0VWZHY2yhjW1/c3KlQ1m+CYEn3UPTbY9wegXEuK64r1
MySjC24cKEXuTR1+hu1yxmVs0DwwfXG/Y1ZrN+jjKaer7xPqphVjP5gRUO4aiu0X
lCbPY5DJPPsvkspjEzFwQQxLhhd6pKjtKstoi1AWVsJHo0EnxHq3gR5LXDIIE/1y
0r6q/lU4tshC3t8T5wQ+xgLmjgO6E3gqbZrW1rj1Oy2aQIEYytzMDWvEGJw6AuiM
ckLKuPSkDmYnFbwEZLehRGqf3K4VHUaYVvDN0uDwHAkGjqvagwnlB8OWKPUNajzX
F9avejMjj6SfqwIDAQABo4IBfDCCAXgwDgYDVR0PAQH/BAQDAgOoMB0GA1UdJQQW
MBQGCCsGAQUFBwMBBggrBgEFBQcDAjAdBgNVHQ4EFgQUNrvcPENu+H1WL2gb7tPM
jTG1PqQwHwYDVR0jBBgwFoAUceFj3mZH65kZ2vmqLG4hKKKQ8LgwcgYIKwYBBQUH
AQEEZjBkMGIGCCsGAQUFBzAChlZodHRwczovL3ZhdWx0LWNsdXN0ZXItcHVibGlj
LXZhdWx0LWI5NGE3YTc5LjFmMjE0YjI5LnoxLmhhc2hpY29ycC5jbG91ZDo4MjAw
L3YxL3BraS9jYTApBgNVHREEIjAggh5pZGVudGl0eS5saW5rZXJkLmNsdXN0ZXIu
bG9jYWwwaAYDVR0fBGEwXzBdoFugWYZXaHR0cHM6Ly92YXVsdC1jbHVzdGVyLXB1
YmxpYy12YXVsdC1iOTRhN2E3OS4xZjIxNGIyOS56MS5oYXNoaWNvcnAuY2xvdWQ6
ODIwMC92MS9wa2kvY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQCxnb2MXdsoQ4Wx96rc
c70ZAUEIbpnfXTF4p43UbTXp2oshu+ZsrMUd+PFmCf669+Y2TGvyen4qYtz0278t
v37urbYk17lqsD1zdg5eQPWeTTjMEzZk4l4/+0xtc3lexYwikTBYbJ6jcYfR7uox
6fOeCITYD1Rvr5EduOB8D52hqdd9CzFBZqyxyR92LB+40XW8R4ty9+Kz7A2ocqqP
KICDIihCIu3cDhHquYibYriOS1TyFMpjcpqCl2/LqceurlC/EFlDMO4FO4LdtYsN
QV8LLoQpWU34pZUTxO/KLGxgsNWWe62pJ90DIcvA8Pdz/cn4POcZ4Y+B475me7Zh
db4J
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgIUJPzHrwe1IJyReqWBSRt5dVVLE7cwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMB4XDTI0MTIzMTIxNTgxN1oX
DTI1MDIwMTIxNTg0N1owGjEYMBYGA1UEAwwPKi5jbHVzdGVyLmxvY2FsMIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyb1lz0sQ4ydZfpJFPgTFTi15yRMI
zQiQwDLHRL32Xd50EOt1DgLrQwB/kQ3jwLFZOTE/bhqp3fwWryVBN3PFqS4IO3ug
n9eHn3GEKS6Ickkcf2oBdVugbL7nfSjG/HsgGIMQwa7xe8pAwiZqNbclt/Z1Cy9G
9OQJV8qoUSqj1ZHHMSzMJ5WMBMTwNJIydc5Kyvkm0zoyVFKUQhw42BF6jZBwynGv
ndlfihBaMeP2GwDIKitpowwdZXEML7jzVhurPsDz1REqTYeAT8rz0GGRz+yHH1tw
CXzTR2yDU8tudIRNk+AMVQnCKDNijeZypw9u+cby5tED6M8F+QijIpNZUQIDAQAB
o38wfTAOBgNVHQ8BAf8EBAMCAQYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU
ceFj3mZH65kZ2vmqLG4hKKKQ8LgwHwYDVR0jBBgwFoAUceFj3mZH65kZ2vmqLG4h
KKKQ8LgwGgYDVR0RBBMwEYIPKi5jbHVzdGVyLmxvY2FsMA0GCSqGSIb3DQEBCwUA
A4IBAQAF9BJ7NXrPmUAQngYU3P3SxjFRGG6nvlBruXeeTOFChi/VFVyN6YKHKpp8
EIqUXIJDr6fofViBPYLTIpkBCDLP5eUl+WdsJgOYjvtGQ3qXjstJykwoNQdrR75I
qqdQ/yyHvyZg/bj9RqgccIGEaTAJuPH+QrzNWqmFVU4j1odyXjQQQqhevr4kz2YN
ho1A3oX4jh9JQhDu/kTfK/bEwKjfksYbby/VNsDTt07ph9wDA3r2r3nRjDTfCIu1
WBgseSdTFgRhaYMjH/5aAP3L0MaXCeuWOkmXMpmc3Ztf5fy/8lbaVYHwMVWzi63V
Xt0ke7nOCdl3fQQU5xTsuglelXgB
-----END CERTIFICATE-----%
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Create linkerd namespace using this command **kubectl create namespace linkerd**

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl create namespace linkerd

namespace/linkerd created
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Step 6: Create a Secret for the Issuer Certificate

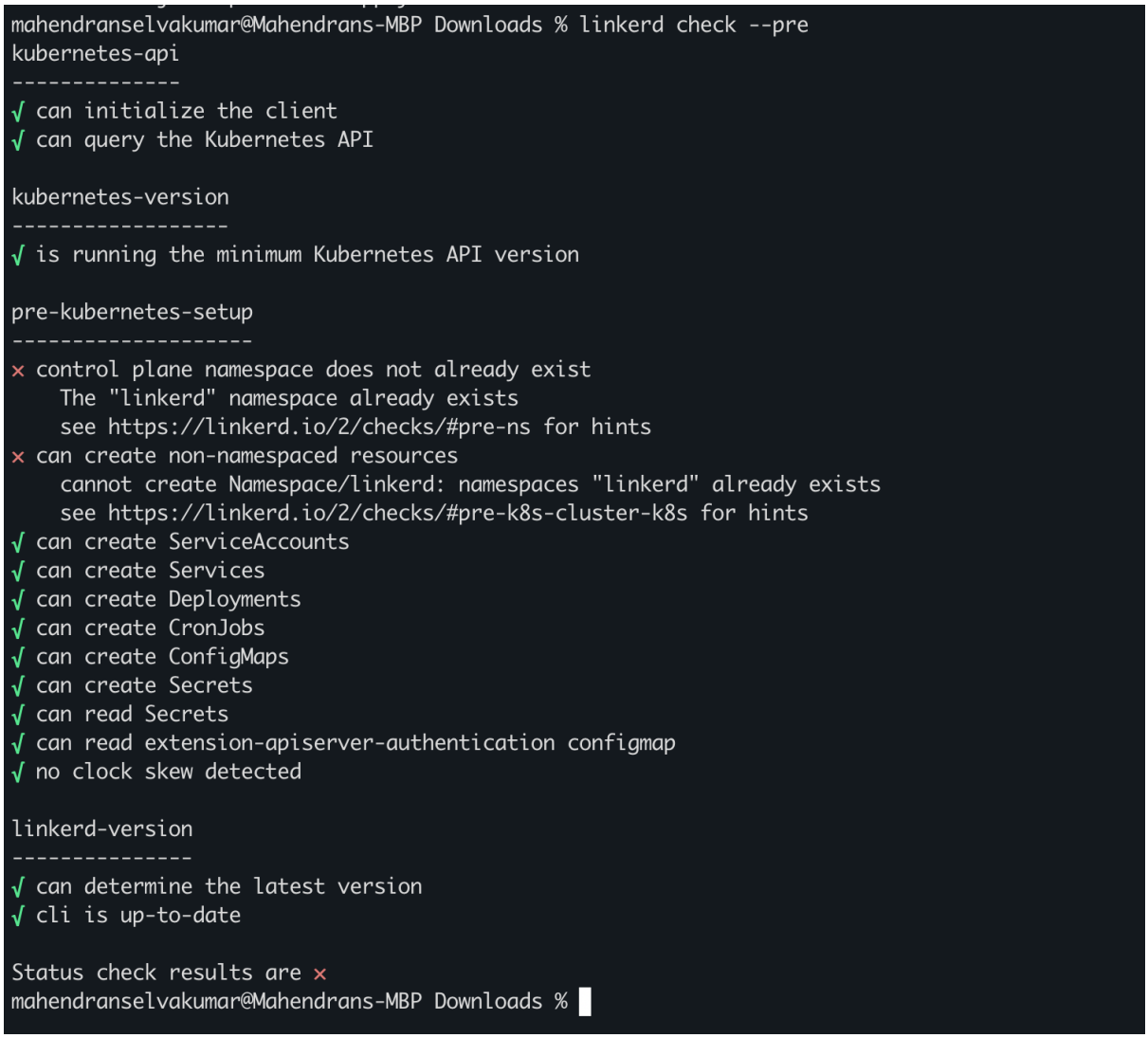
Run this command **kubectl create secret tls linkerd-identity-issuer --cert=linkerd-issuer.pem --key=private-key.pem --namespace=linkerd** to create linkerd-identity-issuer

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl create secret tls linkerd-identity-issuer \
    --cert=linkerd-issuer.pem \
    --key=private-key.pem \
    --namespace=linkerd
secret/linkerd-identity-issuer created
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Run this command **kubectl create configmap linkerd-trust-anchor --from-file=ca.crt=linkerd-cert.pem --namespace=linkerd** to create linkerd-trust-anchor

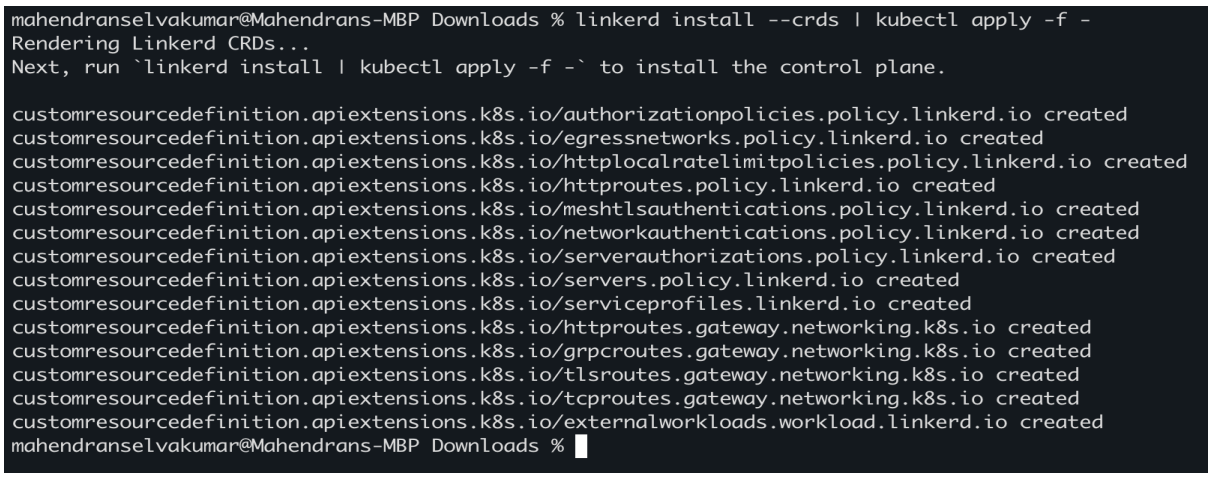
```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl create configmap linkerd-trust-anchor \
    --from-file=ca.crt=linkerd-cert.pem \
    --namespace=linkerd

configmap/linkerd-trust-anchor created
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Run the pre-check command **linkerd check –pre** to ensure your Kubernetes cluster meets Linkerd's requirements. Here you will get namespace already exist checks failed



Step 7: Install Linkerd on your EKS cluster

Run this command **linkerd install --crds | kubectl apply -f -** to install Linkerd's Custom Resource Definitions (CRDs),which must be installed before the control plan



Install the Linkerd control plane using this command **linkerd install | kubectl apply -f -**

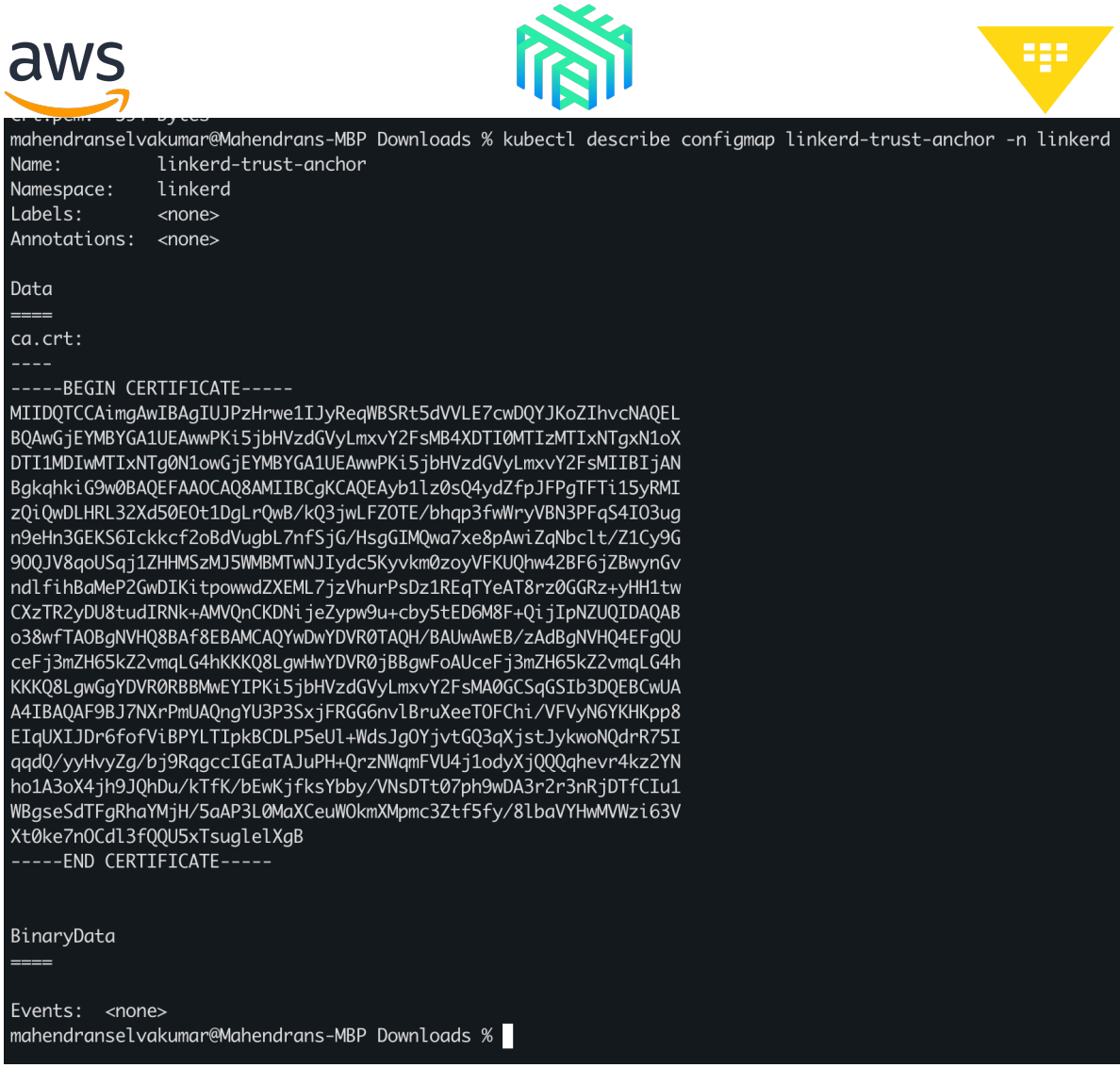
```
mahendranselvakumar@Mahendrans-MBP Downloads % linkerd install | kubectl apply -f -
Warning: resource namespaces/linkerd is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on re
sources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
namespace/linkerd configured
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-identity created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-identity created
serviceaccount/linkerd-identity created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-destination created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-destination created
serviceaccount/linkerd-destination created
secret/linkerd-sp-validator-k8s-tls created
validatingwebhookconfiguration.admissionregistration.k8s.io/linkerd-sp-validator-webhook-config created
secret/linkerd-policy-validator-k8s-tls created
validatingwebhookconfiguration.admissionregistration.k8s.io/linkerd-policy-validator-webhook-config created
clusterrole.rbac.authorization.k8s.io/linkerd-policy created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-destination-policy created
role.rbac.authorization.k8s.io/remote-discovery created
rolebinding.rbac.authorization.k8s.io/linkerd-destination-remote-discovery created
role.rbac.authorization.k8s.io/linkerd-heartbeat created
rolebinding.rbac.authorization.k8s.io/linkerd-heartbeat created
clusterrole.rbac.authorization.k8s.io/linkerd-heartbeat created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-heartbeat created
serviceaccount/linkerd-heartbeat created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-proxy-injector created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-proxy-injector created
serviceaccount/linkerd-proxy-injector created
secret/linkerd-proxy-injector-k8s-tls created
mutatingwebhookconfiguration.admissionregistration.k8s.io/linkerd-proxy-injector-webhook-config created
configmap/linkerd-config created
role.rbac.authorization.k8s.io/ext-namespace-metadata-linkerd-config created
Warning: resource secrets/linkerd-identity-issuer is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only
be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
secret/linkerd-identity-issuer configured
configmap/linkerd-identity-trust-roots created
service/linkerd-identity created
service/linkerd-identity-headless created
deployment.apps/linkerd-identity created
service/linkerd-dst created
service/linkerd-dst-headless created
service/linkerd-sp-validator created
service/linkerd-policy created
service/linkerd-policy-validator created
deployment.apps/linkerd-destination created
cronjob.batch/linkerd-heartbeat created
deployment.apps/linkerd-proxy-injector created
service/linkerd-proxy-injector created
secret/linkerd-config-overrides created
mahendranselvakumar@Mahendrans-MBP Downloads %
```

Step 8: Verify Configuration

Use this command to inspect the details of a Kubernetes Secret named linkerd-identity-issuer in the linkerd namespace: **kubectl describe secret linkerd-identity-issuer -n linkerd**

```
mahendranselvakumar@Mahendrans-MBP Downloads % kubectl describe secret linkerd-identity-issuer -n linkerd
Name:         linkerd-identity-issuer
Namespace:    linkerd
Labels:       linkerd.io/control-plane-component=identity
              linkerd.io/control-plane-ns=linkerd
Annotations:  linkerd.io/created-by: linkerd/cli edge-24.11.8

Type:  kubernetes.io/tls

Data
====
key.pem:  226 bytes
tls.crt:  2741 bytes
tls.key:  1674 bytes
crt.pem:  594 bytes
```

Use this command to inspect the details of a Kubernetes ConfigMap named linkerd-trust-anchor in the linkerd namespace: **kubectl describe configmap linkerd-trust-anchor -n linkerd**



Run a health check to ensure Linkerd is properly configured using this command **linkerd check**

Conclusion

Using HCP Vault Dedicated with Linkerd provides a secure and efficient way to manage certificates and protect communication between services in your Kubernetes cluster. HCP Vault handles certificate issuance and renewal automatically, reducing manual effort and ensuring certificates are always up to date. Linkerd uses these certificates to enable secure, encrypted communication with mTLS, adding an extra layer of protection.

This combination simplifies security management and gives you tools like real-time metrics and traffic monitoring to keep your microservices running smoothly. Together, HCP Vault and Linkerd make it easier to build and maintain a secure, reliable, and scalable Kubernetes environment.

https://www.linkedin.com/in/mahendran-selvakumar/


Keep Learning, Keep Mesh-ing!!

Feel free to reach out to me, if you have any other queries or suggestions Stay connected on LinkedIn: Mahendran Selvakumar

Stay connected on Medium: https://devopstronaut.com/