

## ¿Cómo autentica AWS por CLI?

- Tener los perfiles configurados en \$HOME/.aws/credentials, no es suficiente. En caso de que esos perfiles sean del SSO, haber hecho el login, tampoco es suficiente:

```
SSFF-00004:~ emanuelquiroga$ aws sso login --profile sso
Attempting to automatically open the SSO authorization page in your default browser.
If the browser does not open or you wish to use a different device to authorize this request,
please visit the following URL:

https://device.sso.us-east-1.amazonaws.com/

Then enter the code:

VFWQ-JFBZ
Successfully logged into Start URL: https://naranjax.awsapps.com/start
SSFF-00004:~ emanuelquiroga$ aws iam list-account-aliases
Unable to locate credentials. You can configure credentials by running "aws configure".
SSFF-00004:~ emanuelquiroga$
```

- Una forma de pasarle credenciales al CLI es exportando nuestro “access key id”, “secret access key” y “session token” (si corresponde) a las variables de entorno AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY y AWS\_SESSION\_TOKEN:

```
SSFF-00004:~ emanuelquiroga$ export AWS_ACCESS_KEY_ID="ASIAY4BCJSY3JNURECJV"
SSFF-00004:~ emanuelquiroga$ export AWS_SECRET_ACCESS_KEY="aMv1Z0i0ux5KE3Wqn6CQoqGE8wy5o9C
SSFF-00004:~ emanuelquiroga$ export AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjEP////////wEaCXV
AEaDDYw0Tk1NzI4NzQ3OCIM/NXfmVasfH4AVk2VKvQBFL6qY9T1ko8G3U75mjR5Wtf/dhsRCx0fabzyoY8occRUeYL
iWaY169qNEio9XLS0tZKgXhEtuLFcU29fKHQeGVGULbfVyBwNuZ9nvhZo0K56oURpla/Upamuunv9VadrSZkmP6kDG
5MR4+tHK6cIM1HqPPeIgI11uu0dwd3oqsITwwCjSL4G1ATFI0HU6LRS8V6AiVsMfsxW65UMJ0jQUgsQFgJ0xAYjo5+
XJyEMXWf4SC1taHg18b+aPY/AcrtBv6dljRje2zwfkFfUG51oopIUHBhA9xth9dB2WNe/ZvZTNNLZ9LVw=="
SSFF-00004:~ emanuelquiroga$ aws iam list-account-aliases
{
  "AccountAliases": [
    "nx-sbx"
  ]
}
SSFF-00004:~ emanuelquiroga$
```

- La otra forma es indicarle al CLI qué profile del \$HOME/.aws/credentials utilizar. Esto se puede hacer de varias maneras, la que yo recomiendo es exportando el perfil a la variable de entorno AWS\_PROFILE. **Pero cuidado**, si hay credenciales en AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY y AWS\_SESSION\_TOKEN. **Éstas van a tener SIEMPRE la prioridad**. En la foto se ve cómo exporto el perfil de la cuenta NX-Security y me sigue devolviendo el nombre la cuenta NX-Sandbox:

```
SSFF-00004:~ emanuelquirolga$ export AWS_PROFILE=admin/security
SSFF-00004:~ emanuelquirolga$ aws iam list-account-aliases
{
  "AccountAliases": [
    "nx-sbx"
  ]
}
SSFF-00004:~ emanuelquirolga$ █
```

- Vacío las variables AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY y AWS\_SESSION\_TOKEN... Y ahora sí, me devuelve el nombre de la cuenta, del perfil que exporté a AWS\_PROFILE:

```
SSFF-00004:~ emanuelquirolga$ export AWS_ACCESS_KEY_ID=""
SSFF-00004:~ emanuelquirolga$ echo $AWS_ACCESS_KEY_ID

SSFF-00004:~ emanuelquirolga$ export AWS_SECRET_ACCESS_KEY=""
SSFF-00004:~ emanuelquirolga$ export AWS_SESSION_TOKEN=""
SSFF-00004:~ emanuelquirolga$ aws iam list-account-aliases
{
  "AccountAliases": [
    "nx-security"
  ]
}
SSFF-00004:~ emanuelquirolga$ █
```

- Otra forma de especificarle al CLI de qué perfil tomar las credenciales es agregándole al comando el flag “--profile” con el nombre del perfil a utilizar:

```
SSFF-00004:~ emanuelquirolga$ aws iam list-account-aliases --profile admin/gitlab-prod
{
  "AccountAliases": [
    "nx-gitlab-prod"
  ]
}
SSFF-00004:~ emanuelquirolga$
```

- Por último, si el perfil en el file de credentials se llama “[default]” no es necesario el flag “--profile “ para utilizarlo, **pero tampoco lo recomiendo, ya que si hay otro perfil en AWS\_PROFILE, éste va a tener prioridad y puede generar confusión.**

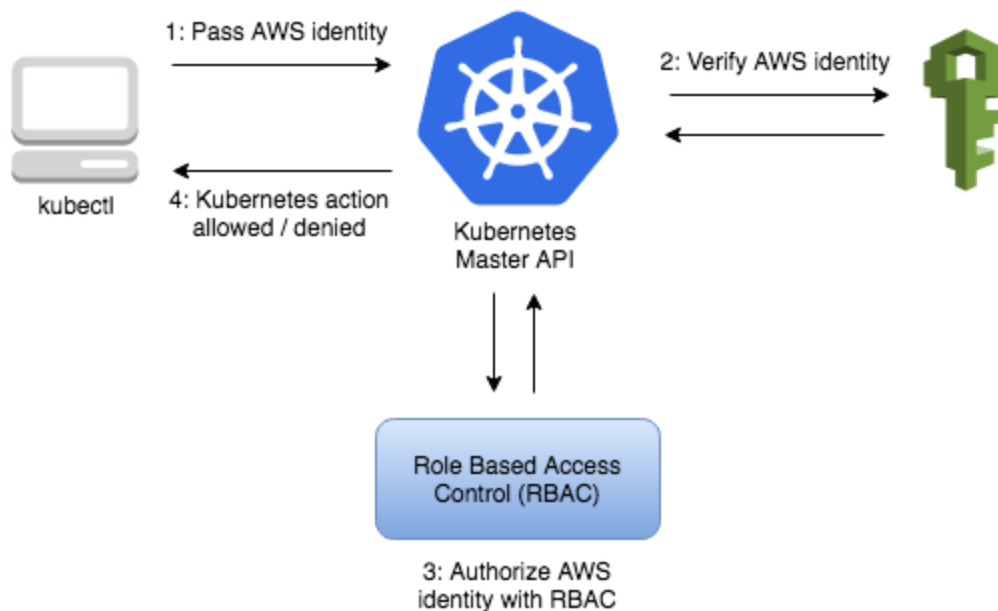
## ¿Cómo autentica EKS?

Partiendo del concepto de que EKS es el servicio de Kubernetes administrado por AWS, esto implica que haya una integración entre la autenticación de Kubernetes y la de AWS.

- En Kubernetes el que hace las autenticaciones es el RBAC (Role Binding Access Control)
- En AWS el que hace las autenticaciones es IAM (Identity Access Management)

En EKS esto está integrado por el Configmap aws-auth, que funciona de la siguiente manera:

Con cada llamada a la API de Kubernetes que uno hace a través de kubectl (o el que usen) primero valida las credenciales de AWS y después valida las credenciales del usuario interno de Kubernetes:



Dónde están todas especificadas todas estas credenciales? En el Kubeconfig del cluster que estemos utilizando... Al hacer :

```
$ aws eks update-kubeconfig --region us-east-1 --name nombreCluster  
--kubeconfig $HOME/.kube/nombreKubeconfig --profile awsProfile
```

EKS les descarga, en el file que hayan indicado al flag “--kubeconfig”, las configuraciones del cluster indicado en el flag “--name” y su certificado para autenticarse. Así como también las credenciales de AWS para autenticarse que hayan especificado en el flag “--profile”

### Aclaraciones:

- Si no especifican un file en `--kubeconfig` EKS buscará uno por default, en primera instancia usará el file que hayan exportado a la variable de entorno `KUBECONFIG` y si no hay ningún file en esa variable de entorno utilizará `$HOME/.kube/config`
- Si no especifican un perfil de AWS con el flag `--profile` buscará credenciales preseteadas, ya sea en las variables de entorno `AWS_ACCESS_KEY_ID`, etc. La variable de entorno `AWS_PROFILE` o un perfil de nombre `"[default]"` .. Respetando las prioridades ya mencionadas anteriormente.
- Si por error creamos un `kubeconfig` con las configuraciones de un cluster, pero las credenciales de otro usuario de AWS, no se va a poder autenticar.

Así queda el Kubeconfig con todas las configuraciones mencionadas:

```
SSFF-00004:kube emanuelquiros@cat nx-new-dev-eks
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUcSUzJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQRBTbkJna3Foa2IHOXcwQkFRc0ZB
TURFeE56RTBOVGt4TTFFvd0ZURVRNqkvHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVuc0lFBGdnRVBBRENDQVFvQ2dnRUJBTGou
dGg5ZWxkcmVudWw1SW5UUDJscampGa3pkTnhjUFc1eEwrddjy4d0hRT0ZQb3Zwb3g0bmhzRzhjdmdJUaVVXMUtWSDDuUTNTVQpZUHkZ0DV4SHZJYZdHNDU1NVd3VWZj
bENlM1JScStKUS9tQzVCOFVvUzYyT21NYVI2ZS5t4dUetK9sUXRFYUwwWnJTVEcKSUX2QjZmWfBnWkxyWlh1bFI2aVfvRWpmV0N1d0xZNXhZZ2dsclJCc21aWE1Q
QVFIL0JBUBURBZ0trTUE4R0ExVWRfD0VCC193UUZNUQ1CQWY4d0RRWUpLb1pjaHZkTkFRRUxUCUFEZ2dFQkfGb2Y1YWtiMkcxYZWRVRXFUTmc3QWVHeXNLRW8KL2Ra
dAp5dVMZbnppPUFVVQVWmazcz0SHZDTzQvUEJUOUJJenptVzhwbXAYtk1Mb1FITmhqK3RSK11MbFgxQkdLVldkdFlTCm5aZ0d0TmlS0VduUGs10WNZclhCc1NiZF
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
server: https://D7435739CA12FE65814D4CC7828E4396.sk1.us-east-1.eks.amazonaws.com
name: arn:aws:eks:us-east-1:735348611665:cluster/nx-new-dev-eks
contexts:
- context:
    cluster: arn:aws:eks:us-east-1:735348611665:cluster/nx-new-dev-eks
    user: arn:aws:eks:us-east-1:735348611665:cluster/nx-new-dev-eks
    name: arn:aws:eks:us-east-1:735348611665:cluster/nx-new-dev-eks
current-context: arn:aws:eks:us-east-1:735348611665:cluster/nx-new-dev-eks
kind: Config
preferences: {}
users:
- name: arn:aws:eks:us-east-1:735348611665:cluster/nx-new-dev-eks
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
        - --region
        - us-east-1
        - eks
        - get-token
        - --cluster-name
        - nx-new-dev-eks
      command: aws
      env:
        - name: AWS_PROFILE
          value: DONT_MODIFY_eks/dev
```

Una vez creados los Kubeconfig (**Yo recomiendo tener uno por cluster**, aunque es posible tener más de un cluster configurado en el mismo file e ir cambiando el entorno por default con kubectl) la forma de autenticarse en ese cluster es exportando la ubicación del file a la variable de entorno KUBECONFIG

```
SSFF-00004:~ emanuelquirolga$ kubectl get nodes -A
The connection to the server localhost:8080 was refused - did you specify the right host or port?
SSFF-00004:~ emanuelquirolga$ export KUBECONFIG=$HOME/.kube/nx-new-dev-eks
SSFF-00004:~ emanuelquirolga$ kubectl get nodes -A
NAME                                STATUS    ROLES    AGE    VERSION
ip-172-18-0-34.ec2.internal         Ready    <none>    90d    v1.14.8-eks-b8860f
ip-172-18-1-107.ec2.internal        Ready    <none>    90d    v1.14.8-eks-b8860f
ip-172-18-3-44.ec2.internal         Ready    <none>    90d    v1.14.8-eks-b8860f
SSFF-00004:~ emanuelquirolga$
```

Tambien se puede usar el file por default, es decir, si tienen el kubeconfig en \$HOME/.kube/config , no es necesario exportarlo a la variable KUBECONFIG, ya que ese es el file por default que busca Kubernetes... **De todas formas, tampoco lo recomiendo ya que es fácil confundir sobre qué cluster estás parado y corres el riesgo de realizar en un cluster lo que querías hacer en otro.**

**Aclaracion:** Si tienen un Kubeconfig exportado en la variable KUBECONFIG, éste va a tener prioridad por sobre el default