Creating and managing clusters

Creating a cluster

Create a simple cluster with the following command:

```
eksctl create cluster
```

That will create an EKS cluster in your default region (as specified by your AWS CLI configuration) with one managed nodegroup containing two m5.large nodes.

Note

eksctl now creates a managed nodegroup by default when a config file isn't used. To create a self-managed nodegroup, pass `--managed=false` to `eksctl create cluster` or `eksctl create nodegroup`.

Note

In `us-east-1` you are likely to get `UnsupportedAvailabilityZoneException`. If you do, copy the suggested zones and pass `--zones` flag, e.g. `eksctl create cluster --region=us-east-1 --zones=us-east-1a,us-east-1b,us-east-1d`. This may occur in other regions, but less likely. You shouldn't need to use `--zone` flag otherwise.

After the cluster has been created, the appropriate kubernetes configuration will be added to your kubeconfig file. This is, the file that you have configured in the environment variable `KUBECONFIG` or `~/.kube/config` by default. The path to the kubeconfig file can be overridden using the `--kubeconfig` flag.

Other flags that can change how the kubeconfig file is written:

flag	type	use	default value
--kubeconfig	string	path to write kubeconfig (incompatible with –auto-kubeconfig)	$KUBECONFIG or ~/.kube/config
--set-kubeconfig-context	bool	if true then current-context will be set in kubeconfig; if a context is already set then it will be overwritten	true
--auto-kubeconfig	bool	save kubeconfig file by cluster name	true
--write-kubeconfig	bool	toggle writing of kubeconfig	true

Using Config Files

You can create a cluster using a config file instead of flags.

First, create `cluster.yaml` file:

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: basic-cluster
  region: eu-north-1

nodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 10
    volumeSize: 80
    ssh:
      allow: true # will use ~/.ssh/id_rsa.pub as the default ssh key
  - name: ng-2
    instanceType: m5.xlarge
    desiredCapacity: 2
    volumeSize: 100
    ssh:
      publicKeyPath: ~/.ssh/ec2_id_rsa.pub
```

Next, run this command:

```
eksctl create cluster -f cluster.yaml
```

This will create a cluster as described.

If you needed to use an existing VPC, you can use a config file like this:

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-in-existing-vpc
  region: eu-north-1

vpc:
  subnets:
    private:
      eu-north-1a: { id: subnet-0ff156e0c4a6d300c }
      eu-north-1b: { id: subnet-0549cdab573695c03 }
      eu-north-1c: { id: subnet-0426fb4a607393184 }

nodeGroups:
  - name: ng-1-workers
    labels: { role: workers }
    instanceType: m5.xlarge
    desiredCapacity: 10
    privateNetworking: true
  - name: ng-2-builders
    labels: { role: builders }
    instanceType: m5.2xlarge
    desiredCapacity: 2
    privateNetworking: true
    iam:
      withAddonPolicies:
        imageBuilder: true
```

Note

The cluster name or nodegroup name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 128 characters otherwise you will get a validation error. More information can be found here

To delete this cluster, run:

```
eksctl delete cluster -f cluster.yaml
```

Note

Without the `--wait` flag, this will only issue a delete operation to the cluster's CloudFormation stack and won't wait for its deletion.

In some cases, AWS resources using the cluster or its VPC may cause cluster deletion to fail. To ensure any deletion errors are propagated in `eksctl delete cluster`, the `--wait` flag must be used. If your delete fails or you forget the wait flag, you may have to go to the CloudFormation GUI and delete the eks stacks from there.

Note

When deleting a cluster with nodegroups, in some scenarios, Pod Disruption Budget (PDB) policies can prevent nodes from being removed successfully from nodepools. E.g. a cluster with `aws-ebs-csi-driver` installed, by default, spins off two pods while having a PDB policy that allows at most one pod to be unavailable at a time. This will make the other pod unevictable during deletion. To successfully delete the cluster, one should use `disable-nodegroup-eviction` flag. This will bypass checking PDB policies.

```
eksctl delete cluster -f cluster.yaml --disable-nodegroup-eviction
```

See `examples/` directory for more sample config files.

Dry Run

The dry-run feature enables generating a ClusterConfig file that skips cluster creation and outputs a ClusterConfig file that represents the supplied CLI options and contains the default values set by eksctl.

More info can be found on the Dry Run page.

Was this page helpful?

👍 👎