



kainlite

Mendoza, Argentina

Search

- Home
- Blog
- Categories
- Tags
- About
- Preferences

From zero to hero with kops and AWS

2019-01-19 linux kubernetes AWS kops kubernetes Comments Word Count: 3334 words Read Count: 16 minutes

Introduction

In this article we will create a cluster from scratch with [kops](#) (K8s installation, upgrades and management) in [AWS](#), We will configure [aws-alb-ingress-controller](#) (External traffic into our services/pods) and [external dns](#) (Update the records based in the ingress rules) and also learn a bit about awscli in the process.

Basically we will have a fully functional cluster that will be able to handle public traffic in minutes, first we will install the cluster with kops, then we will enable the ingress controller and lastly external-dns, then we will deploy a basic app to test that everything works fine, SSL/TLS is out of the scope but it's fairly easy to implement if you are using ACM.

Just in case you don't know this setup is not going to be free, cheap for sure because we will use small instances, etc, but not completely free, so before you dive in, be sure that you can spend a few bucks testing it out.

Kops

This is an awesome tool to setup and maintain your clusters, currently only compatible with AWS and GCE, other platforms are planned and some are also supported in alpha, we will be using AWS in this example, it requires kubectl so make sure you have it installed:

```
curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | jq -r .tag_name) kops-linux-amd64
chmod +x kops-linux-amd64
sudo mv kops-linux-amd64 /usr/local/bin/kops
```

Export the credentials that we will be using to create the kops user and policies

```
export AWS_ACCESS_KEY_ID=XXXX && export AWS_SECRET_ACCESS_KEY=XXXXX
```

You can do it this way or just use `aws configure` and set a profile.

The next thing that we need are IAM credentials for kops to work, you will need awscli configured and working with your AWS admin-like account most likely before proceeding:

```
# Create iam group
aws iam create-group --group-name kops
# OUTPUT:
# {
#   "Group": {
#     "Path": "/",
#     "GroupName": "kops",
#     "GroupId": "AGPAIABI3O4WYM46AIX44",
#     "Arn": "arn:aws:iam::894527626897:group/kops",
#     "CreateDate": "2019-01-18T01:04:23Z"
#   }
# }

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess --group kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonRoute53FullAccess --group kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess --group kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/IAMFullAccess --group-name kops

# Attach policies
aws iam create-user --user-name kops
aws iam add-user-to-group --user-name kops --group-name kops
# OUTPUT:
# {
#   "Group": {
#     "Path": "/",
#     "GroupName": "kops",
#     "GroupId": "AGPAIABI3O4WYM46AIX44",
#     "Arn": "arn:aws:iam::894527626897:group/kops",
#     "CreateDate": "2019-01-18T01:04:23Z"
#   }
# }

# Create access key - save the output of this command.
aws iam create-access-key --user-name kops
# OUTPUT:
# {
#   "AccessKey": {
#     "UserName": "kops",
#     "AccessKeyId": "AKIAJE*****",
#     "Status": "Active",
#     "SecretAccessKey": "zWJhfemER*****",
#     "CreateDate": "2019-01-18T01:05:44Z"
#   }
# }
```

The last command will output the access key and the secret key for the `kops` user, save that information because we will use it from now on, note that we gave kops a lot of power with that user, so be careful with the keys.

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster

2022-10-25

Running Rust on ARM32v7 via QEMU.

2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.

2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).

2021-07-22

Kubernetes image policy webhook explained

2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)





kainlite

Mendoza, Argentina

Search

- Home
- Blog
- Categories
- Tags
- About
- Preferences

Additional permissions to be able to create ALBs

```
cat << EOF > kops-alb-policy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "iam:CreateServiceLinkedRole",
        "tag:GetResources",
        "elasticloadbalancing:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
EOF

aws iam create-policy --policy-name kops-alb-policy --policy-document file://kops-alb-policy
# OUTPUT:
# {
#   "Policy": {
#     "PolicyName": "kops-alb-policy",
#     "PolicyId": "ANPAIRIYZZZTCPJGNZZXS",
#     "Arn": "arn:aws:iam::894527626897:policy/kops-alb-policy",
#     "Path": "/",
#     "DefaultVersionId": "v1",
#     "AttachmentCount": 0,
#     "PermissionsBoundaryUsageCount": 0,
#     "IsAttachable": true,
#     "CreateDate": "2019-01-18T03:50:00Z",
#     "UpdateDate": "2019-01-18T03:50:00Z"
#   }
# }

cat << EOF > kops-route53-policy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets"
      ],
      "Resource": [
        "arn:aws:route53::hostedzone/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ListHostedZones",
        "route53:ListResourceRecordSets"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
EOF

aws iam create-policy --policy-name kops-route53-policy --policy-document file://kops-route53-policy
# OUTPUT:
# {
#   "Policy": {
#     "PolicyName": "kops-route53-policy",
#     "PolicyId": "ANPAIEWAGN62HBYC7QOS2",
#     "Arn": "arn:aws:iam::894527626897:policy/kops-route53-policy",
#     "Path": "/",
#     "DefaultVersionId": "v1",
#     "AttachmentCount": 0,
#     "PermissionsBoundaryUsageCount": 0,
#     "IsAttachable": true,
#     "CreateDate": "2019-01-18T03:15:37Z",
#     "UpdateDate": "2019-01-18T03:15:37Z"
#   }
# }
```

Note that even we just created these kops policies for alb and route53 we cannot add them right now, we need to first create the cluster, you can skip them if you don't plan on using these resources.

Now we will also export or set the cluster name and kops state store as environment variables

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster

2022-10-25

Running Rust on ARM32v7 via QEMU.

2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.

2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).

2021-07-22

Kubernetes image policy webhook explained

2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)

```
export NAME=k8s.techsquad.rocks
export KOPS_STATE_STORE=techsquad-cluster-state-store
```

We will be using these in a few places, so to not repeat ourselves let's better have it as variables.

Create the zone for the subdomain in Route53

```
ID=$(uuidgen) && aws route53 create-hosted-zone --name ${NAME} --caller-reference $ID | jq .
# OUTPUT:
# [
#   "ns-848.awsdns-42.net",
#   "ns-12.awsdns-01.com",
#   "ns-1047.awsdns-02.org",
#   "ns-1862.awsdns-40.co.uk"
# ]
```

As I'm already using this domain for the blog with github we can create a subdomain for it and add some NS records in our root zone for that subdomain, in this case k8s.techsquad.rocks. To make this easier I will show you how it should look like:

<input type="checkbox"/>	NS	k8s.techsquad.rocks	ns-848.awsdns-42.net	300	N/A	Edit	Delete	Created: 2019-01-17
<input type="checkbox"/>	NS	k8s.techsquad.rocks	ns-12.awsdns-01.com	300	N/A	Edit	Delete	Created: 2019-01-17
<input type="checkbox"/>	NS	k8s.techsquad.rocks	ns-1047.awsdns-02.org	300	N/A	Edit	Delete	Created: 2019-01-17
<input type="checkbox"/>	NS	k8s.techsquad.rocks	ns-1862.awsdns-40.co.uk	300	N/A	Edit	Delete	Created: 2019-01-17

So with this change and our new zone in Route53 for the subdomain, we can freely manage it like if it was another domain, this means that everything that goes to *.k8s.techsquad.rocks will be handled by our Route53 zone.

Create a bucket to store the cluster state

```
aws s3api create-bucket \
  --bucket ${KOPS_STATE_STORE} \
  --region us-east-1
# OUTPUT:
# {
#   "Location": "/techsquad-cluster-state-store"
# }
```

Note that bucket names are unique, so it's always a good idea to prefix them with your domain name or something like that.

Set the versioning on, in case we need to rollback at some point

```
aws s3api put-bucket-versioning --bucket ${KOPS_STATE_STORE} --versioning-configuration Sta
```

Set encryption on for the bucket

```
aws s3api put-bucket-encryption --bucket ${KOPS_STATE_STORE} --server-side-encryption-confi
```

And finally let's create our cluster

```
export KOPS_STATE_STORE="s3://${KOPS_STATE_STORE}"

kops create cluster \
  --zones us-east-1a \
  --networking calico \
  ${NAME} \
  --yes
# OUTPUT:
# I0117 23:14:06.449479 10314 create_cluster.go:1318] Using SSH public key: /home/kainlite
# I0117 23:14:08.367862 10314 create_cluster.go:472] Inferred --cloud=aws from zone "us-ea
# I0117 23:14:09.736030 10314 subnets.go:184] Assigned CIDR 172.20.32.0/19 to subnet us-ea
# W0117 23:14:18.049687 10314 firewall.go:249] Opening etcd port on masters for access fr
# I0117 23:14:19.385541 10314 executor.go:91] Tasks: 0 done / 77 total; 34 can run
# I0117 23:14:21.779681 10314 vfs_castore.go:731] Issuing new certificate: "apiserver-aggr
# I0117 23:14:21.940026 10314 vfs_castore.go:731] Issuing new certificate: "ca"
# I0117 23:14:24.404810 10314 executor.go:91] Tasks: 34 done / 77 total; 24 can run
# I0117 23:14:26.548234 10314 vfs_castore.go:731] Issuing new certificate: "master"
# I0117 23:14:26.689470 10314 vfs_castore.go:731] Issuing new certificate: "apiserver-aggr
# I0117 23:14:26.766563 10314 vfs_castore.go:731] Issuing new certificate: "kube-schedule
# I0117 23:14:26.863562 10314 vfs_castore.go:731] Issuing new certificate: "kube-controlle
# I0117 23:14:26.955776 10314 vfs_castore.go:731] Issuing new certificate: "kubecfg"
# I0117 23:14:26.972837 10314 vfs_castore.go:731] Issuing new certificate: "apiserver-prox
# I0117 23:14:26.973239 10314 vfs_castore.go:731] Issuing new certificate: "kops"
# I0117 23:14:27.055466 10314 vfs_castore.go:731] Issuing new certificate: "kubelet"
# I0117 23:14:27.127778 10314 vfs_castore.go:731] Issuing new certificate: "kubelet-api"
# I0117 23:14:27.570516 10314 vfs_castore.go:731] Issuing new certificate: "kube-proxy"
# I0117 23:14:29.503168 10314 executor.go:91] Tasks: 58 done / 77 total; 17 can run
```

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster

2022-10-25

Running Rust on ARM32v7 via QEMU.

2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.

2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).

2021-07-22

Kubernetes image policy webhook explained

2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)



kainlite

Mendoza, Argentina

Search

- Home
- Blog
- Categories
- Tags
- About
- Preferences

```
# I0117 23:14:31.594404 10314 executor.go:91] Tasks: 75 done / 77 total; 2 can run
# I0117 23:14:33.297131 10314 executor.go:91] Tasks: 77 done / 77 total; 0 can run
# I0117 23:14:33.297168 10314 dns.go:153] Pre-creating DNS records
# I0117 23:14:34.947302 10314 update_cluster.go:291] Exporting kubecfg for cluster
# kops has set your kubectl context to k8s.techsquad.rocks
#
# Cluster is starting. It should be ready in a few minutes.
#
# Suggestions:
# * validate cluster: kops validate cluster
# * list nodes: kubectl get nodes --show-labels
# * ssh to the master: ssh -i ~/.ssh/id_rsa admin@api.k8s.techsquad.rocks
# * the admin user is specific to Debian. If not using Debian please use the appropriate user
# * read about installing addons at: https://github.com/kubernetes/kops/blob/master/docs/
```

We set the KOPS_STATE_STORE to a valid S3 url for kops, and then created the cluster, this will set kubectl context to our new cluster, we might need to wait a few minutes before being able to use it, but before doing anything let's validate that's up and ready.

```
kops validate cluster ${NAME}
# OUTPUT:
# Using cluster from kubectl context: k8s.techsquad.rocks
#
# Validating cluster k8s.techsquad.rocks
#
# INSTANCE GROUPS
# NAME                                ROLE    MACHINETYPE    MIN    MAX    SUBNETS
# master-us-east-1a                  Master  m3.medium      1      1      us-east-1a
# nodes                              Node    t2.medium      2      2      us-east-1a
#
# NODE STATUS
# NAME                                ROLE    READY
# ip-172-20-39-123.ec2.internal      node    True
# ip-172-20-52-65.ec2.internal      node    True
# ip-172-20-61-51.ec2.internal      master  True
#
# Your cluster k8s.techsquad.rocks is ready
```

The validation passed and we can see that our cluster is ready, it can take several minutes until the cluster is up and functional, in this case it took about 3-5 minutes.

We will create an additional subnet to satisfy our ALB:

```
aws ec2 create-subnet --vpc-id vpc-06e2e104ad785474c --cidr-block 172.20.64.0/19 --availability-zone us-east-1a
# OUTPUT:
# {
#   "Subnet": {
#     "AvailabilityZone": "us-east-1b",
#     "AvailableIpAddressCount": 8187,
#     "CidrBlock": "172.20.64.0/19",
#     "DefaultForAz": false,
#     "MapPublicIpOnLaunch": false,
#     "State": "pending",
#     "SubnetId": "subnet-017a5609ce6104e1b",
#     "VpcId": "vpc-06e2e104ad785474c",
#     "AssignIpv6AddressOnCreation": false,
#     "Ipv6CidrBlockAssociationSet": []
#   }
# }

aws ec2 create-tags --resources subnet-017a5609ce6104e1b --tags Key=KubernetesCluster,Value=k8s.techsquad.rocks
aws ec2 create-tags --resources subnet-017a5609ce6104e1b --tags Key=Name,Value=us-east-1b.k8s.techsquad.rocks
aws ec2 create-tags --resources subnet-017a5609ce6104e1b --tags Key=SubnetType,Value=Public
aws ec2 create-tags --resources subnet-017a5609ce6104e1b --tags Key=kubernetes.io/cluster/k8s.techsquad.rocks
aws ec2 create-tags --resources subnet-017a5609ce6104e1b --tags Key=kubernetes.io/role/elb,Value=controller
```

Note that we applied some required tags for the controller, and created an extra subnet, in a HA setup this would not be necessary since kops would create it for us but this is a small testing/dev cluster, so we will need to do it manually.

And lastly a security group for our ALB:

```
aws ec2 create-security-group --group-name WebApps --description "Default web security group"
# OUTPUT:
# {
#   "GroupId": "sg-09f0b1233696e65ef"
# }

aws ec2 authorize-security-group-ingress --group-id sg-09f0b1233696e65ef --protocol tcp --port 80 --source 0.0.0.0/0
aws ec2 authorize-security-group-ingress --group-id sg-057d2b0f6e288aa70 --protocol all --port 0 --source 0.0.0.0/0
```

Note that this rule will open the port 80 to the world, you can add your ip or your VPN ips there if you want to restrict it, the second rule will allow the traffic from the load balancer to reach the nodes where our app is running.

Aws-alb-ingress-controller

We will use [Aws ALB Ingress Controller](#), to serve our web traffic, this will create a managed ALB based on our ingress rules.

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster

2022-10-25

Running Rust on ARM32v7 via QEMU.

2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.

2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).

2021-07-22

Kubernetes image policy webhook explained

2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)

```
kubect1 apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/master/docs/docs/clusterrole.rbac.authorization.k8s.io "alb-ingress-controller" created
clusterrolebinding.rbac.authorization.k8s.io "alb-ingress-controller" created
serviceaccount "alb-ingress" created
```

Download the manifest and then modify the cluster-name to `k8s.techsquad.rocks` and a few other parameters, you can list the vpcs with `aws ec2 describe-vpcs` it will have some kops tags, so it's easy to identify.

```
curl -sS "https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/docs/clusterrole.rbac.authorization.k8s.io "alb-ingress-controller" created"
```

Or copy and paste the following lines:

```
cat << EOF > alb-ingress-controller.yaml
# Application Load Balancer (ALB) Ingress Controller Deployment Manifest.
# This manifest details sensible defaults for deploying an ALB Ingress Controller.
# GitHub: https://github.com/kubernetes-sigs/aws-alb-ingress-controller
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: alb-ingress-controller
  name: alb-ingress-controller
# Namespace the ALB Ingress Controller should run in. Does not impact which
# namespaces it's able to resolve ingress resource for. For limiting ingress
# namespace scope, see --watch-namespace.
namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: alb-ingress-controller
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: alb-ingress-controller
    spec:
      containers:
        - args:
            - --v=1
            # Limit the namespace where this ALB Ingress Controller deployment will
            # resolve ingress resources. If left commented, all namespaces are used.
            # - --watch-namespace=your-k8s-namespace
            - --feature-gates=waf=false

            # Setting the ingress-class flag below ensures that only ingress resources with
            # annotation kubernetes.io/ingress.class: "alb" are respected by the controller.
            # choose any class you'd like for this controller to respect.
            - --ingress-class=alb

            # Name of your cluster. Used when naming resources created
            # by the ALB Ingress Controller, providing distinction between
            # clusters.
            - --cluster-name=k8s.techsquad.rocks

            # AWS VPC ID this ingress controller will use to create AWS resources.
            # If unspecified, it will be discovered from ec2metadata.
            - --aws-vpc-id=vpc-06e2e104ad785474c

            # AWS region this ingress controller will operate in.
            # If unspecified, it will be discovered from ec2metadata.
            # List of regions: http://docs.aws.amazon.com/general/latest/gr/rande.html#vpc_1
            - --aws-region=us-east-1

            # Enables logging on all outbound requests sent to the AWS API.
            # If logging is desired, set to true.
            # - ---aws-api-debug
            # Maximum number of times to retry the aws calls.
            # defaults to 10.
            # - --aws-max-retries=10

      env:
        # AWS key id for authenticating with the AWS API.
        # This is only here for examples. It's recommended you instead use
        # a project like kube2iam for granting access.
        #- name: AWS_ACCESS_KEY_ID
        # value: KEYVALUE

        # AWS key secret for authenticating with the AWS API.
        # This is only here for examples. It's recommended you instead use
        # a project like kube2iam for granting access.
        #- name: AWS_SECRET_ACCESS_KEY
```

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster
2022-10-25

Running Rust on ARM32v7 via QEMU.
2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.
2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).
2021-07-22

Kubernetes image policy webhook explained
2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)



kainlite

Mendoza, Argentina

Search

- Home
- Blog
- Categories
- Tags
- About
- Preferences

```
# value: SECRETVALUE
# Repository location of the ALB Ingress Controller.
image: 894847497797.dkr.ecr.us-west-2.amazonaws.com/aws-alb-ingress-controller:v1.1.0
imagePullPolicy: Always
name: server
resources: {}
terminationMessagePath: /dev/termination-log
dnsPolicy: ClusterFirst
restartPolicy: Always
securityContext: {}
terminationGracePeriodSeconds: 30
serviceAccountName: alb-ingress
serviceAccount: alb-ingress

EOF
```

Note that I only modified the args section if you want to compare it with the original.

Then finally apply it.

```
kubectl apply -f alb-ingress-controller.yaml
# OUTPUT:
# deployment.apps "alb-ingress-controller" created
```

External-dns

[External DNS](#) will update our zone in Route53 based in the ingress rules as well, so everything will be done automatically for us once we add an ingress resource.

But first let's attach those policies that we created before:

```
aws iam attach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-route53-policy \
aws iam attach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-route53-policy \
aws iam attach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-alb-policy -- \
aws iam attach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-alb-policy --
```

Note that we just used the policies that we created before but we needed the cluster running because kops creates the roles nodes.k8s.techsquad.rocks and masters.k8s.techsquad.rocks, and this is needed for the aws-alb-ingress-controller and external-dns so these are able to do their job.

We need to download the manifests and modify a few parameters to match our deployment, the parameters are domain-filter and txt-owner-id, the rest is as is:

```
curl -Ss https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.0
```

This configuration will only update records, that's the default policy (upsert), and it will only look for public hosted zones.

Or copy and paste the following lines:

```
cat << EOF > external-dns.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: external-dns
rules:
- apiGroups: [""]
  resources: ["services"]
  verbs: ["get", "watch", "list"]
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["extensions"]
  resources: ["ingresses"]
  verbs: ["get", "watch", "list"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: external-dns-viewer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
- kind: ServiceAccount
  name: external-dns
  namespace: default
---
apiVersion: extensions/v1beta1
```

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster

2022-10-25

Running Rust on ARM32v7 via QEMU.

2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.

2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).

2021-07-22

Kubernetes image policy webhook explained

2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)



kainlite

Mendoza, Argentina

Search

- Home
- Blog
- Categories
- Tags
- About
- Preferences

```
kind: Deployment
metadata:
  name: external-dns
spec:
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: external-dns
    spec:
      serviceAccountName: external-dns
      containers:
        - name: external-dns
          image: registry.opensource.zalan.do/teapot/external-dns:v0.5.9
          args:
            - --source=service
            - --source=ingress
            - --domain-filter=k8s.techsquad.rocks # will make ExternalDNS see only the hosted zones
            - --provider=aws
            - --policy=upsert-only # would prevent ExternalDNS from deleting any records, omit if you want it
            - --aws-zone-type=public # only look at public hosted zones (valid values are public and private)
            - --registry=txt
            - --txt-owner-id=k8s.techsquad.rocks
EOF
```

And apply it:

```
kubectl apply -f external-dns.yaml
# OUTPUT:
# serviceaccount "external-dns" unchanged
# clusterrole.rbac.authorization.k8s.io "external-dns" configured
# clusterrolebinding.rbac.authorization.k8s.io "external-dns-viewer" configured
# deployment.extensions "external-dns" created
```

Validate that we have everything that we installed up and running:

```
kubectl get pods
# OUTPUT:
# NAME                                READY   STATUS    RESTARTS   AGE
# external-dns-7d7998f7bb-lb5kq       1/1     Running   0           2m

kubectl get pods -n kube-system
# OUTPUT:
# NAME                                READY   STATUS    RESTARTS   AGE
# alb-ingress-controller-5885ddd5f9-9rsc8      1/1     Running   0           12m
# calico-kube-controllers-f6bc47f75-n99t1      1/1     Running   0           27m
# calico-node-4ps9c                          2/2     Running   0           25m
# calico-node-kjztv                          2/2     Running   0           27m
# calico-node-zs4fg                          2/2     Running   0           25m
# dns-controller-67f5c6b7bd-r67pl             1/1     Running   0           27m
# etcd-server-events-ip-172-20-42-37.ec2.internal 1/1     Running   0           26m
# etcd-server-ip-172-20-42-37.ec2.internal      1/1     Running   0           26m
# kube-apiserver-ip-172-20-42-37.ec2.internal    1/1     Running   0           27m
# kube-controller-manager-ip-172-20-42-37.ec2.internal 1/1     Running   0           26m
# kube-dns-756bfc7fdf-2kzjs                    3/3     Running   0           24m
# kube-dns-756bfc7fdf-rq5nd                    3/3     Running   0           27m
# kube-dns-autoscaler-787d59df8f-c2d52          1/1     Running   0           27m
# kube-proxy-ip-172-20-42-109.ec2.internal      1/1     Running   0           25m
# kube-proxy-ip-172-20-42-37.ec2.internal       1/1     Running   0           26m
# kube-proxy-ip-172-20-54-175.ec2.internal      1/1     Running   0           25m
# kube-scheduler-ip-172-20-42-37.ec2.internal   1/1     Running   0           26m
```

We can see that alb-ingress-controller is running, also external-dns, and everything looks good and healthy, time to test it with a deployment.

Testing everything

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller
# OUTPUT:
# namespace "2048-game" created
# deployment.extensions "2048-deployment" created
# service "service-2048" created
```

We need to download and edit the ingress resource to make it use our domain so we can then see the record pointing to the ALB.

```
curl -Ss https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0
```

Or just copy and paste the next snippet.

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster

2022-10-25

Running Rust on ARM32v7 via QEMU.

2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.

2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).

2021-07-22

Kubernetes image policy webhook explained

2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- cloud (1)

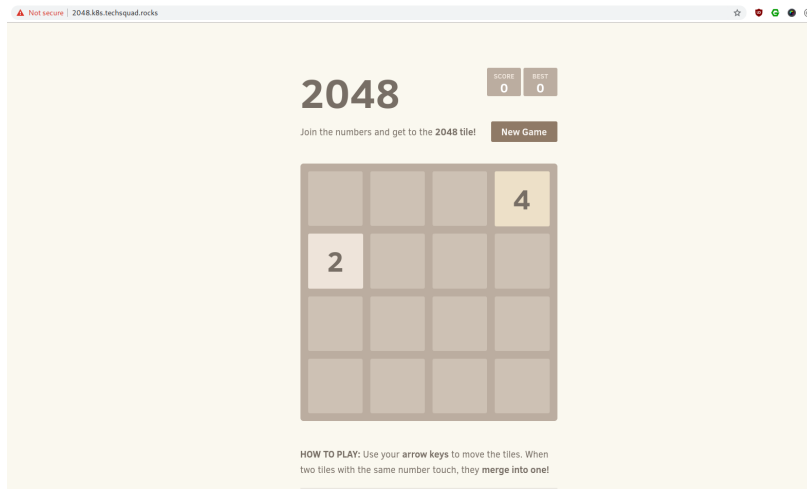


kainlite

Mendoza, Argentina

Search

- Home
- Blog
- Categories
- Tags
- About
- Preferences



Clean up

Remember this is not free, and if you don't want to get charged after you're done testing just shutdown and delete everything.

```
kubectl delete -f 2048-ingress.yaml
aws iam detach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-route53-policy
aws iam detach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-route53-policy
aws iam detach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-alb-policy --i
aws iam detach-role-policy --policy-arn arn:aws:iam::894527626897:policy/kops-alb-policy --i

kops delete cluster ${NAME} --yes
# OUTPUT:
# ...
# Deleted kubectl config for k8s.techsquad.rocks
#
# Deleted cluster: "k8s.techsquad.rocks"
```

This command is really verbose, so I skipped it to the end, be aware that in order to delete the cluster with kops you first need to detach the additionally attached privileges. Also be careful to delete first the ingress resources so the ALB gets removed before you delete the cluster, or you will have an ALB laying around afterwards. You can re-run it if it gets stuck and cannot delete any resource.

Notes

- I was going to use helm and deploy a more complex application here, but the article was already too long, so I decided to go with the aws alb ingress controller example.
- If something doesn't go well or things aren't happening you can always check the logs for external-dns and aws-alb-ingress-controller, the messages are usually very descriptive and easy to understand.
- For an ALB you need two subnets in two different AZs beforehand.
- If you are going to use ALBs, have in mind that it will create an ALB for each deployment, there is a small project that merges everything into one ALB but you need to have a unified or consolidated way to do health checks or or some of the apps will fail and the ALB will return a 502, the project can be found [here](#).
- Documenting what you do and how you do it (Also keeping the documentation updated is really important), not only will help the future you (Yes, you can thank your past self when reading and old doc), but also it will make it easier to share the knowledge and purpose of whatever you are implementing with your team.
- I spent 3 bucks with all the instances and dns zones, etc during this tutorial in case you are interested ;).
- Notes I also removed all \$ from the code blocks and added the output of the commands with # OUTPUT:, let me know if this is clear and easy to read, or if you have any suggestion.

Errata

If you spot any error or have any suggestion, please send me a message so it gets fixed.

Also, you can check the source code and changes in the [generated code](#) and the [sources here](#)

Permalink: https://techsquad.rocks/blog/from_zero_to_hero_with_kops_and_aws/
License: MIT License

kainlite DevOps Practitioner

Articles about Kubernetes, CI/CD, Git, Linux, Containers, Golang, and probably some more random stuff, you can subscribe via RSS or JSON Feed.

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster
2022-10-25

Running Rust on ARM32v7 via QEMU.
2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.
2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).
2021-07-22

Kubernetes image policy webhook explained
2021-01-07

Categories


- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)


Tags


- arm (3)
- aws (5)
- ci/cd (1)


What do you think?


3 Responses


Upvote

Funny

Love


Surprised

Angry

Sad

1 Comment





Login




Join the discussion...


LOG IN WITH


OR SIGN UP WITH DISQUS



Sort by Best







Femi Brand · 2 years ago

Hello good work on this tutorial. I followed your tutorial but Ingress does not create load balancer of my AWS


^


|


▼

· Reply

· Share

Subscribe


Privacy

Do Not Sell My Data

DISQUS

<

>



f

tw

g+

in

Recent Posts

Testing tekton to build and push images for my K3S ARM Oracle cluster
2022-10-25

Running Rust on ARM32v7 via QEMU.
2022-09-03

Running Rust on ARM32v7 K3S Oracle cluster.
2022-09-02

Custom Kubernetes Operator With TypeScript (JavaScript).
2021-07-22

Kubernetes image policy webhook explained
2021-01-07

Categories

- arm (2)
- continuous-integration (2)
- deployment-tools (6)
- kubernetes (14)
- linux (10)
- serverless (10)
- service-mesh (2)
- terraform (1)

Tags

- arm (3)
- aws (5)
- ci/cd (1)