



Experiences with AWS EKS

Kubernetes and Cloud Native Group Dresden

ALFRED KROHMER

LogMeIn[®]
Be Limitless.

Structure

- **Introduction**
 - overview / EKS cluster creation
 - reasons for our ongoing migration (and what we had before)
- **AWS-specific cluster addons**
 - VPC CNI plugin
 - external-dns
 - alb-ingress-controller
 - kube2iam
- **Worker node management**
 - cluster-autoscaler
- **Q&A**

Introduction

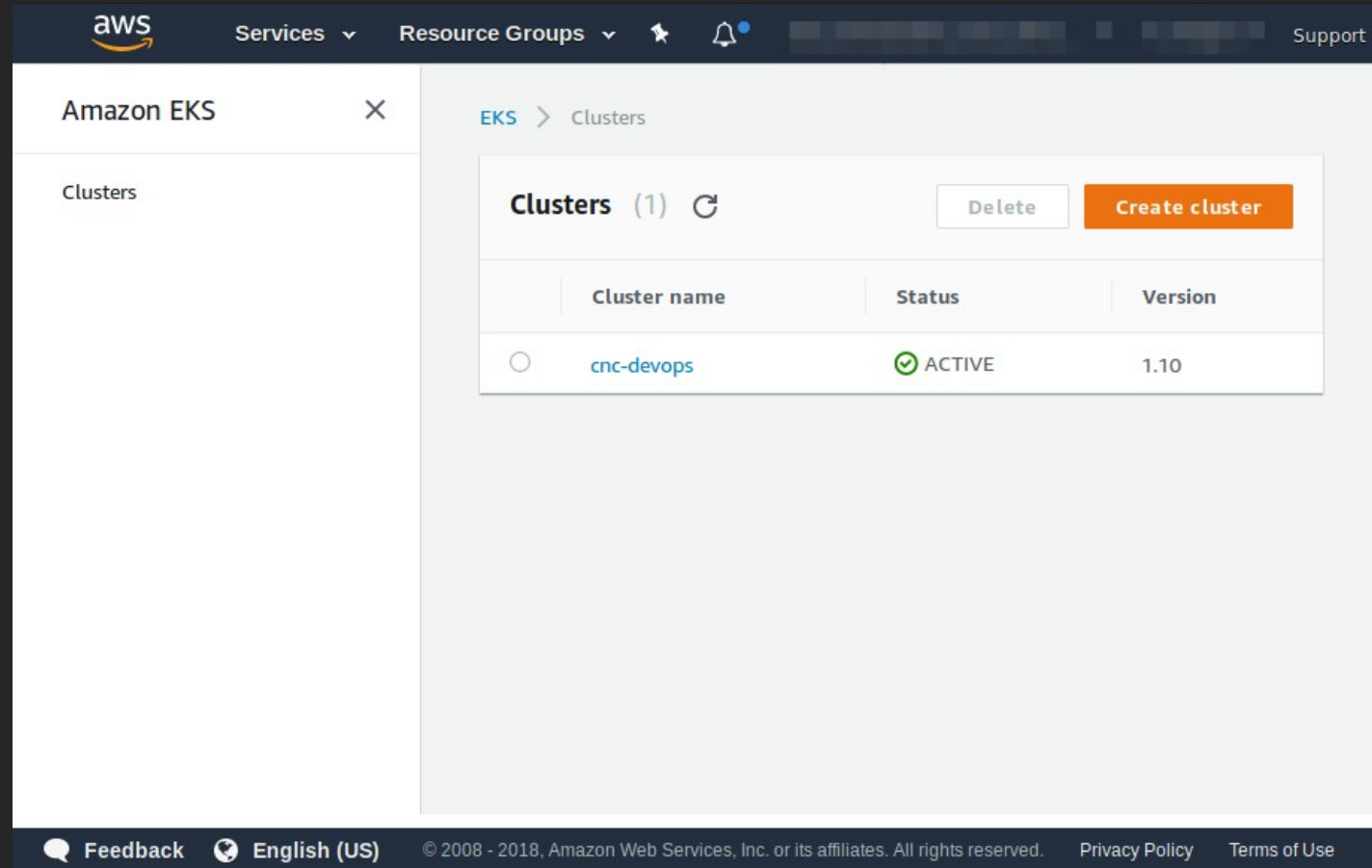
- **Overview**

- EKS = Elastic Kubernetes Service
- AWS provisions and maintains master nodes
- you provision and maintain worker nodes
 - “EKS-optimized” AMI by AWS
- authentication via »AWS IAM Authenticator for Kubernetes«

<https://github.com/kubernetes-sigs/aws-iam-authenticator>

Introduction

- **Cluster overview**



The screenshot shows the Amazon EKS Clusters overview page. The left sidebar has a header 'Amazon EKS' and a 'Clusters' link. The main content area shows a breadcrumb 'EKS > Clusters' and a summary section with 'Clusters (1)' and a refresh icon. There are 'Delete' and 'Create cluster' buttons. Below is a table with one cluster.

	Cluster name	Status	Version
<input type="radio"/>	cnc-devops	✓ ACTIVE	1.10

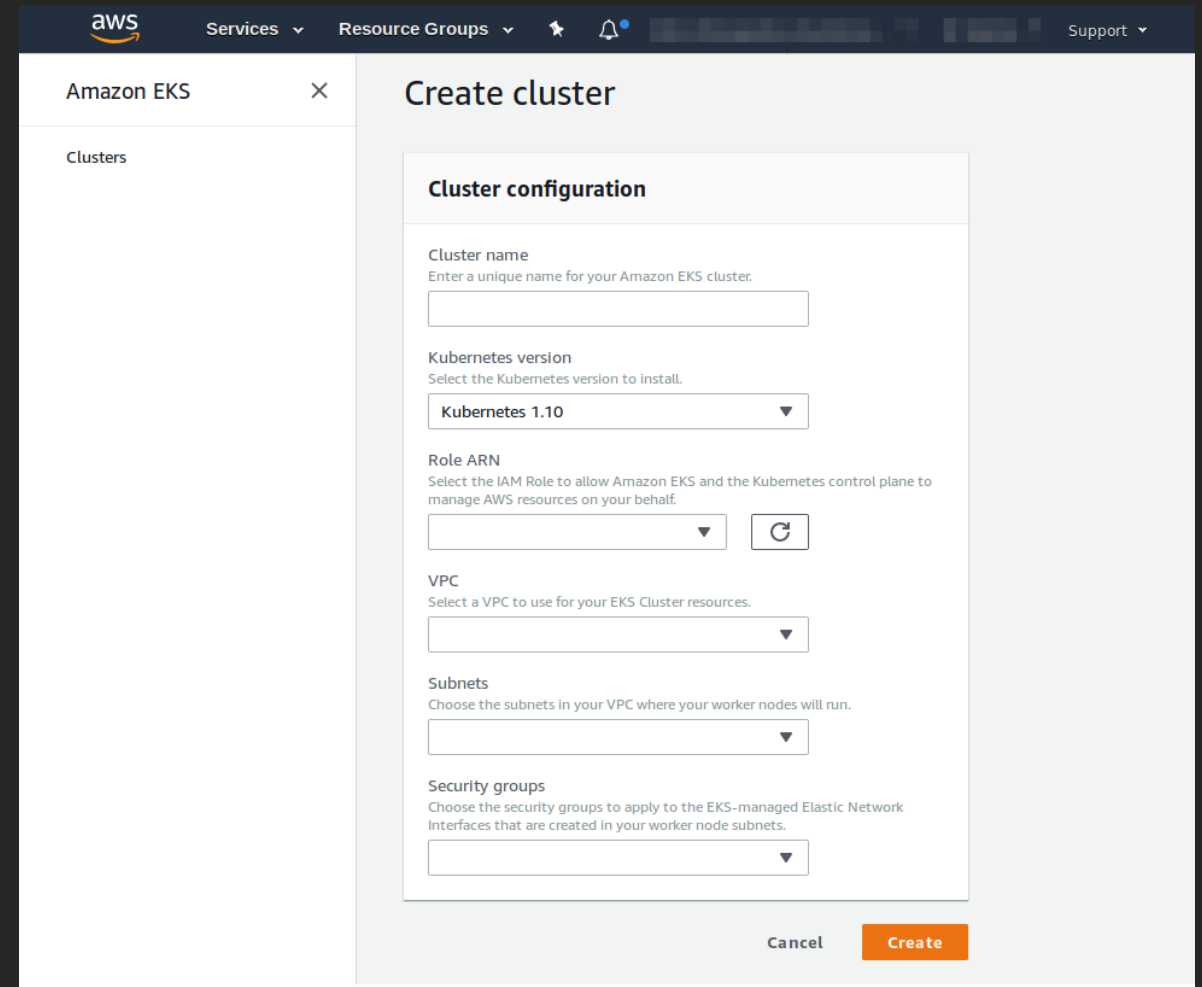
The footer contains 'Feedback', 'English (US)', copyright information '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

Introduction

- **Cluster creation**

IAM user which creates the cluster is (and stays) Kubernetes admin user (for now)

→ use a headless account to create the cluster



The screenshot shows the AWS Management Console interface for creating an Amazon EKS cluster. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a 'Support' link. The left sidebar shows 'Amazon EKS' and 'Clusters'. The main content area is titled 'Create cluster' and contains a 'Cluster configuration' form. The form includes the following fields:

- Cluster name:** A text input field with the instruction 'Enter a unique name for your Amazon EKS cluster.'
- Kubernetes version:** A dropdown menu with the instruction 'Select the Kubernetes version to install.' and the selected value 'Kubernetes 1.10'.
- Role ARN:** A dropdown menu with the instruction 'Select the IAM Role to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf.' and a refresh button.
- VPC:** A dropdown menu with the instruction 'Select a VPC to use for your EKS Cluster resources.'
- Subnets:** A dropdown menu with the instruction 'Choose the subnets in your VPC where your worker nodes will run.'
- Security groups:** A dropdown menu with the instruction 'Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.'

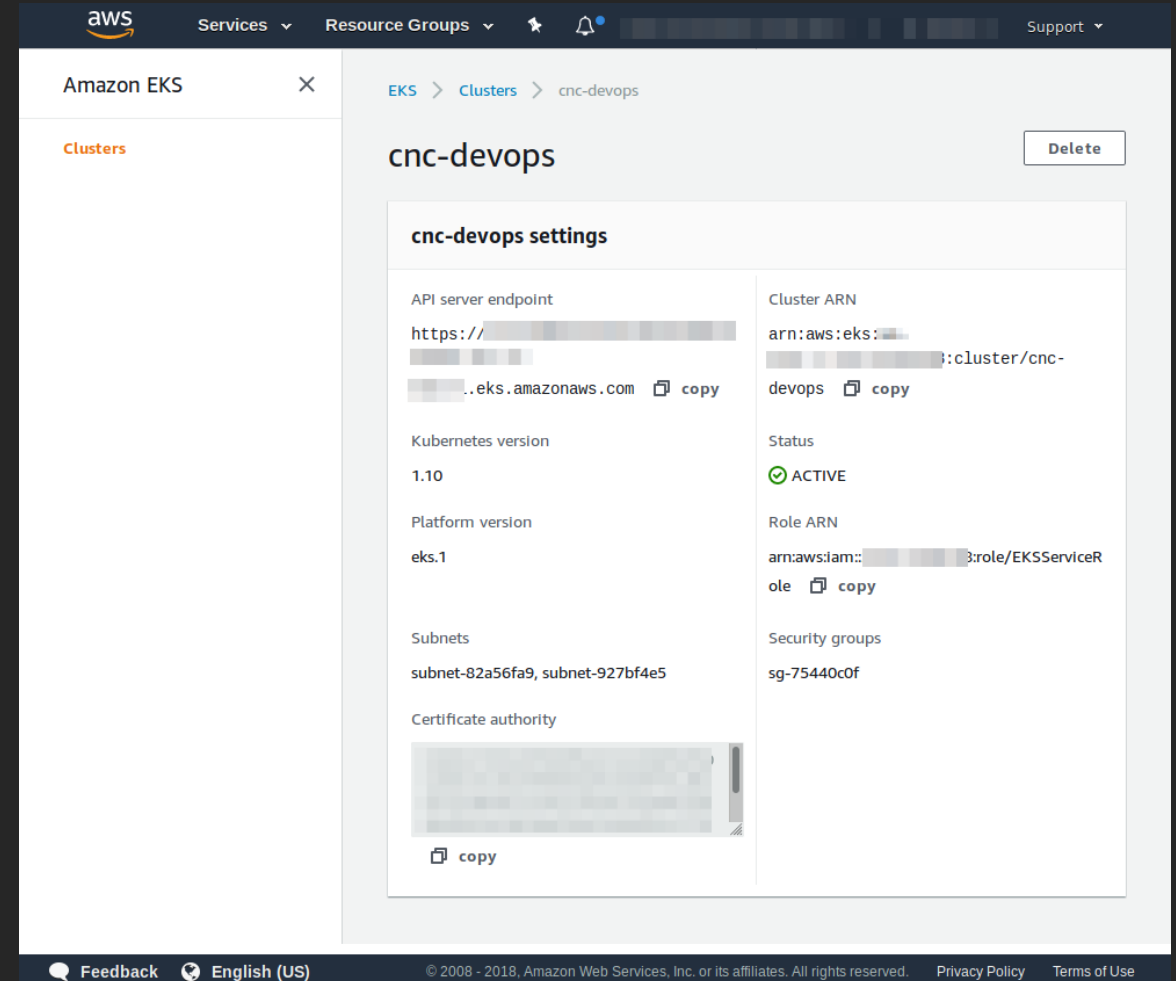
At the bottom right of the form are 'Cancel' and 'Create' buttons.

Introduction

- **Cluster details**

→ details cannot be modified (for now)

→ “Platform versions” are supposed to be rolled out by AWS



The screenshot displays the AWS Management Console interface for an Amazon EKS cluster. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a 'Support' dropdown. The left sidebar shows 'Amazon EKS' with a 'Clusters' link. The main content area is titled 'cnc-devops' and includes a 'Delete' button. Below the title is a 'cnc-devops settings' section with a table of cluster details:

API server endpoint	Cluster ARN
https://[redacted].eks.amazonaws.com	arn:aws:eks:[redacted]:[redacted]:cluster/cnc-devops
Kubernetes version	Status
1.10	ACTIVE
Platform version	Role ARN
eks.1	arn:aws:iam::[redacted]:role/EKSServiceRole
Subnets	Security groups
subnet-82a56fa9, subnet-927bf4e5	sg-75440c0f
Certificate authority	
[redacted]	

Each row in the table includes a 'copy' icon for copying the value. The bottom of the console shows a footer with 'Feedback', 'English (US)', and copyright information: '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

Introduction

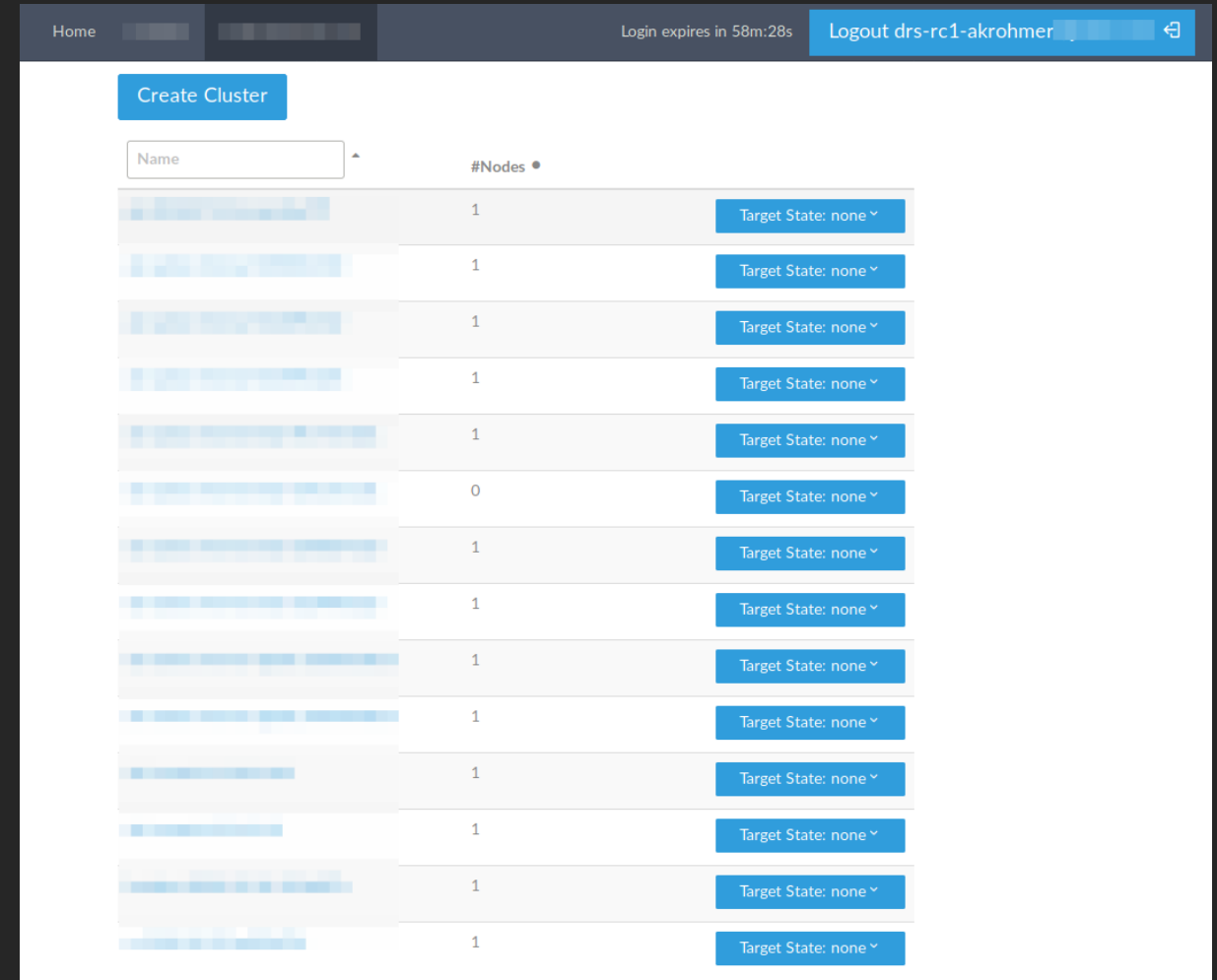
- **Our (ongoing) migration away from**
 - **AWS ECS:**
 - slow (especially with CloudFormation)
 - lack of flexibility
 - vendor lock-in
 - EC2 instances unmaintained
 - Fargate much more expensive than EC2 spot
 - 30s maximum graceful termination period
 - cluster needs to have enough capacity **before** scheduling a task / service

Introduction

- **Our (ongoing) migration away from**
 - **our own container orchestration framework**
 - availability problems (in dev)
 - lack of flexibility
 - one “pod” per host deployments only
 - overly complicated setup with a custom host agent and in-container agent both connecting to ZooKeeper, with frontend UI and REST API
 - starting containers
 - providing application configuration

Introduction

- **Our (ongoing) migration away from**
 - **our own container orchestration framework**



Introduction

- **Our (ongoing) migration away from**
 - **our own container orchestration framework**
- all nodes hosted via EC2

The screenshot displays the LogMeIn AWS EKS console interface. At the top, there's a navigation bar with 'Home', 'Login expires in 56m:3s', and a 'Logout' button. The main content area is divided into three tabs: 'Node', 'Configuration', and 'Resolved Configuration'. The 'Node' tab is active, showing a table of target states for Instance, Bundle, and Host. All three are in a 'Running' state. Below this, the 'Instance' section provides details like cloudType (EC2), datacenter, domain, fqdn, hostn, instanceId, instanceName, instanceType (c4.2xlarge), location (us-east-1c), software, softwareVersion, and zooKeeperRootPrefix (/fleet/rc/). The 'Instance status' section shows additionalStateInformation (Services pending: None), appliedConfigVersion (0), pendingConfigVersion (0), currentState (running), and targetState (running). A table below lists services, their URIs, additional state information (active:0, pending:0), and status (running). The 'Host' section shows host information like hostname, version (2.10), type (c4.2xlarge), location, tags, and Name. The 'Host status' section shows version (2.10), hostname, currentState (running), and bundleTargetState (running).

AWS-specific cluster addons

- **VPC CNI plugin**
 - CNI plugin for kubelet
 - open source
 - can be used without EKS / with custom Kubernetes installations in EC2
 - <https://github.com/aws/amazon-vpc-cni-k8s>

AWS-specific cluster addons

- **VPC CNI plugin**
 - uses AWS VPC as “overlay” network for Kubernetes
 - assigns secondary IPs to EC2 worker nodes
 - routing rules pass traffic through veth pairs to pod network
 - all other EC2 instances in your VPC can connect to your pods directly

AWS-specific cluster addons

- **VPC CNI plugin**

- beware of the defaults:
 - allocates as many IPs as possible per instance
→ IP starvation in small subnets
 - secondary network interfaces: uses SNAT to primary private IP to reach outside of VPC
- “number of available IPs” is not managed as a resource

AWS-specific cluster addons

- **external-dns**
 - creates Route 53 resource records for your services, ingresses and StatefulSet (or single) pods

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    external-dns.alpha.kubernetes.io/hostname: nginx.external-dns-test.my-org.com.
spec:
  type: LoadBalancer
  ports:
    - port: 80
      name: http
      targetPort: 80
  selector:
    app: nginx
```

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: foo
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
    - host: foo.bar.com
      http:
        paths:
          - backend:
              serviceName: foo
              servicePort: 80
```

- <https://github.com/kubernetes-incubator/external-dns/blob/master/docs/tutorials/aws.md>

AWS-specific cluster addons

- **external-dns**
 - uses Route 53 TXT records for own storage
 - also supports other cloud providers
 - missing feature for us:
creating per-pod DNS records

AWS-specific cluster addons

- **alb-ingress-controller**
 - creates AWS Application Load Balancers (ALBs) for your ingresses

targets are the VPC IPs of your pods

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: foo
  annotations:
    kubernetes.io/ingress.class: "alb"
spec:
  rules:
    - host: foo.bar.com
      http:
        paths:
          - backend:
              serviceName: foo
              servicePort: 80
```

Listener Rule

Target group

- <https://github.com/kubernetes-sigs/aws-alb-ingress-controller>

AWS-specific cluster addons

- **alb-ingress-controller**
 - stops “controlling” an ALB if you add non-existing services
 - can bring down other services behind the same ALB

AWS-specific cluster addons

- **kube2iam**
 - deployed as daemon set
 - intercepts traffic to EC2 metadata service at `http://169.254.169.254/` via iptables and serves its own metadata service for pods
 - <https://github.com/jtblin/kube2iam>

```
apiVersion: v1
kind: Pod
metadata:
  name: aws-cli
  labels:
    name: aws-cli
  annotations:
    iam.amazonaws.com/role: role-arn
spec:
  containers:
    - image: fstab/aws-cli
      command:
        - "/home/aws/aws/env/bin/aws"
        - "s3"
        - "ls"
        - "some-bucket"
      name: aws-cli
```

AWS-specific cluster addons

- **kube2iam**
 - allows to assign IAM roles to your pods
 - whitelist allowed pod roles per namespace via annotation
 - EC2 instance role needs to be allowed to assume all pod roles

Worker node management

- **provisioning auto scaling groups (ASGs)**
 - via CloudFormation
 - one stack per ASG
 - one ASG per
 - instance type
 - launch type (on-demand / spot)
 - availability zone
 - service type (RTC / non-RTC, special kernel settings)
- huge matrix of ASGs

Worker node management

- **provisioning auto scaling groups (ASGs)**
 - description of the matrix of all required combinations in YAML file
 - read and excuted by python script
 - still missing:
 - automated roll-over of instances
 - automated deletion of unneeded ASGs

Worker node management

- **cluster-autoscaler**
 - another cluster-addon
 - supports multiple cloud providers
 - scales ASGs up & down according to number of pending pods and underutilized nodes
 - can be installed per-namespace to support fine-granular scaling behavior
- <https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler>

Worker node management

- **cluster-autoscaler**
 - beware of the defaults:
 - stops working for some time if some actions fail
 - can scale up all ASGs to their maximum number of instances
 - don't deploy it on spot instances!

The background of the entire image is a photograph of a sunset. The sky is filled with horizontal bands of orange, yellow, and light brown clouds. In the foreground, the dark silhouettes of evergreen trees are visible against the bright sky. On the left side, there is a semi-transparent grey rectangular box.

Q&A

Thank you.

