Berlin | November 20 - 21, 2018

{codemotion}

AWS EKS & beyond

Master your Kubernetes deployment on AWS

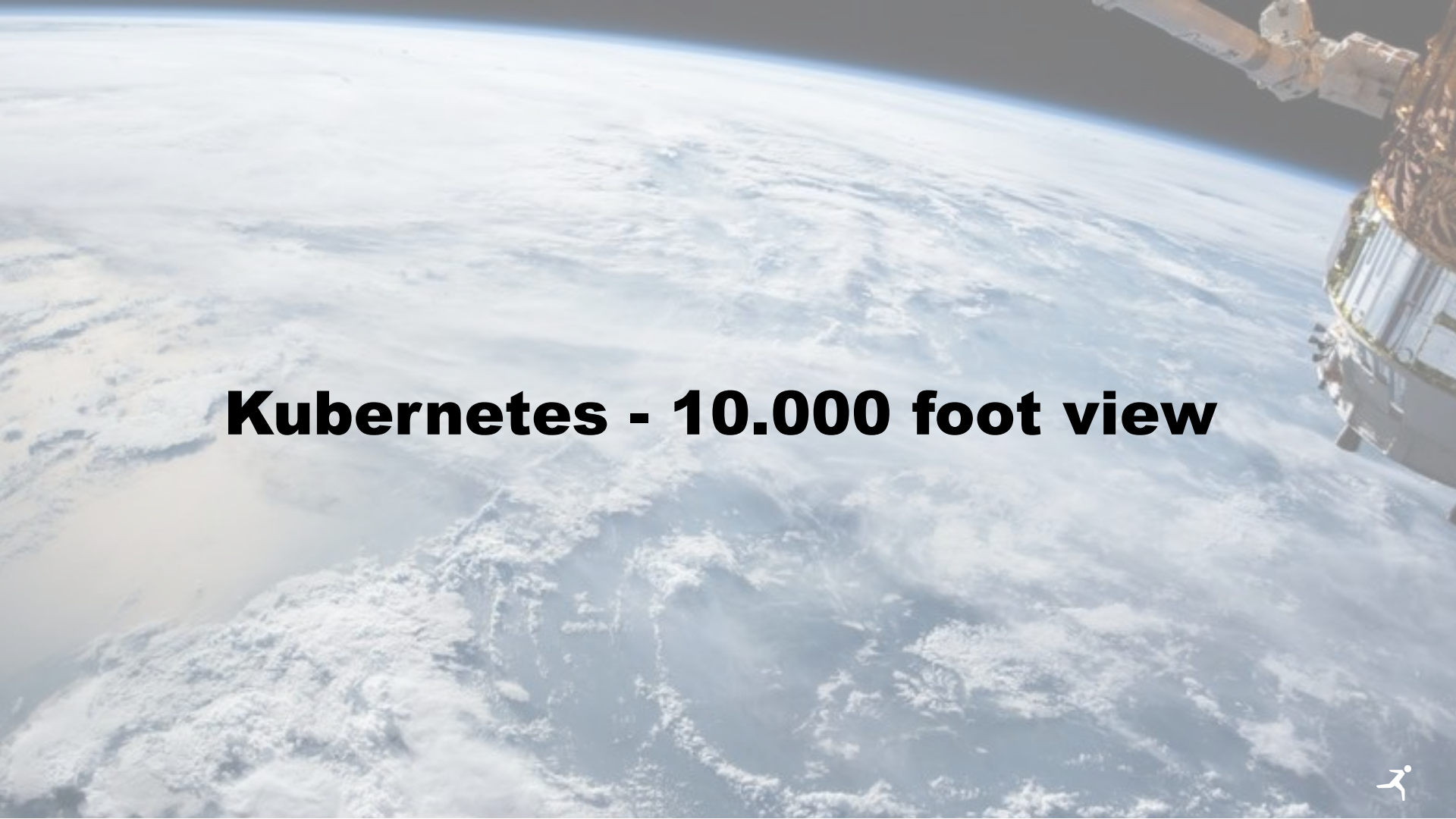
Max Körbächer

Hey!

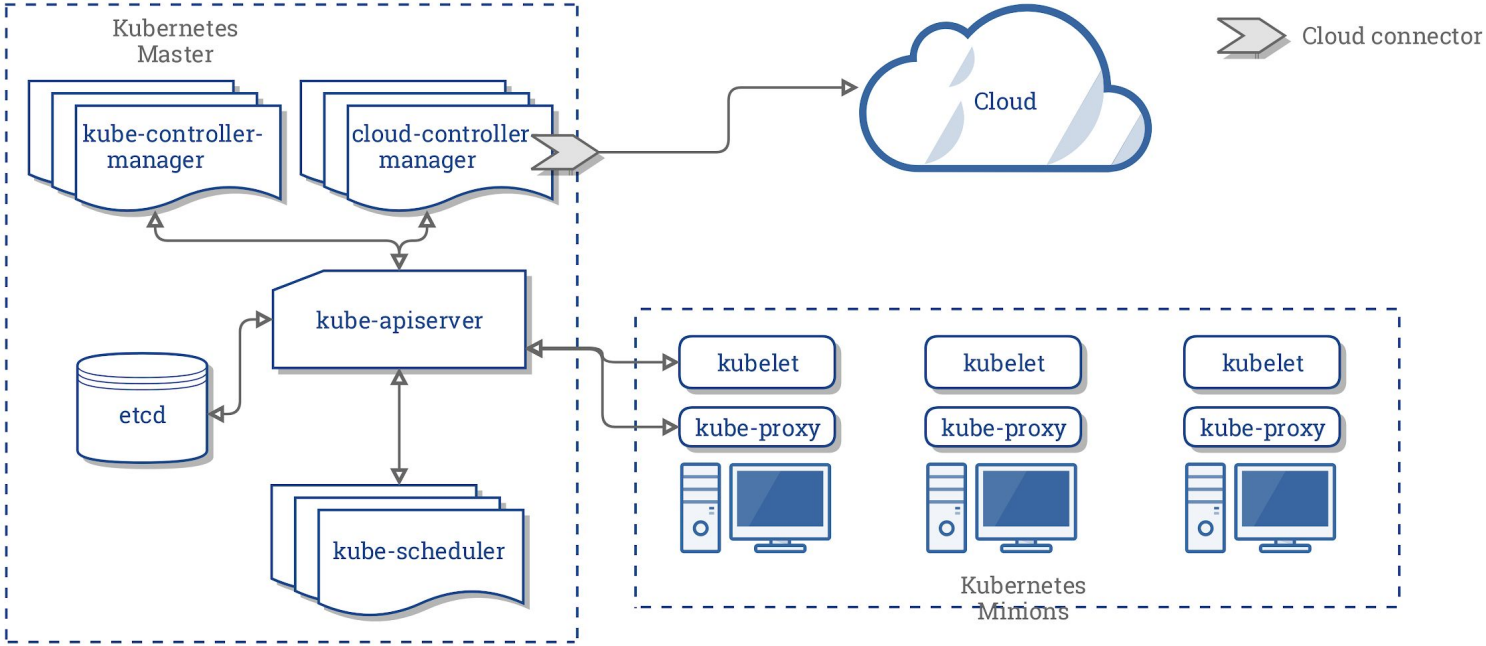
Max Körbächer

Cloud Solution Architect @ Storm Reply

- **Design and build cloud ready solutions**
  - microservice & event driven apps
  - serverless & kubernetes based
  - ♥ for GO, GraphQL & NoSQL
- **Background as Enterprise Architect & Founder**
- **Visit me at: max.koerbaecher.io**


Kubernetes - 10.000 foot view

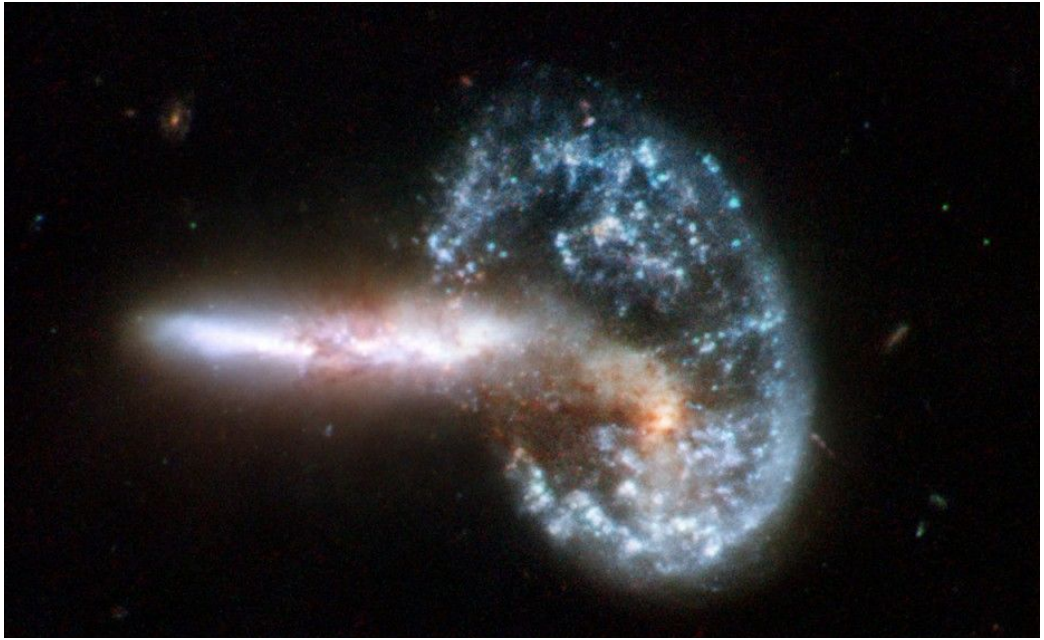
K8s foundation ...



source:https://kubernetes.io/docs/concepts/architecture/cloud-controller/

K8s & the cloud

Two concepts collide into each other



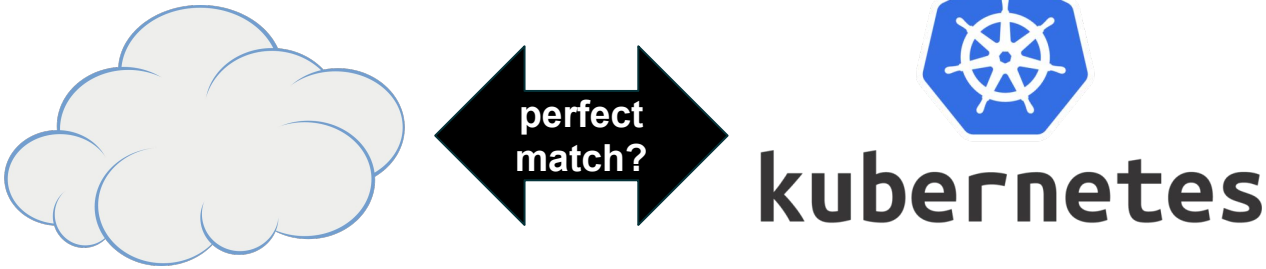
source: http://annesastronomynews.com

In theory K8s & cloud is a dream

For a perfect K8s cluster you need:

- auto scaling server
- software defined storage
- redundancy / high availability
- managed databases
- reliable and fast file storage

However…on the fine grained level there are might be some differences which you get to feel the more complex you make your cluster




Amazon EKS
-
Elastic Container Service for Kubernetes

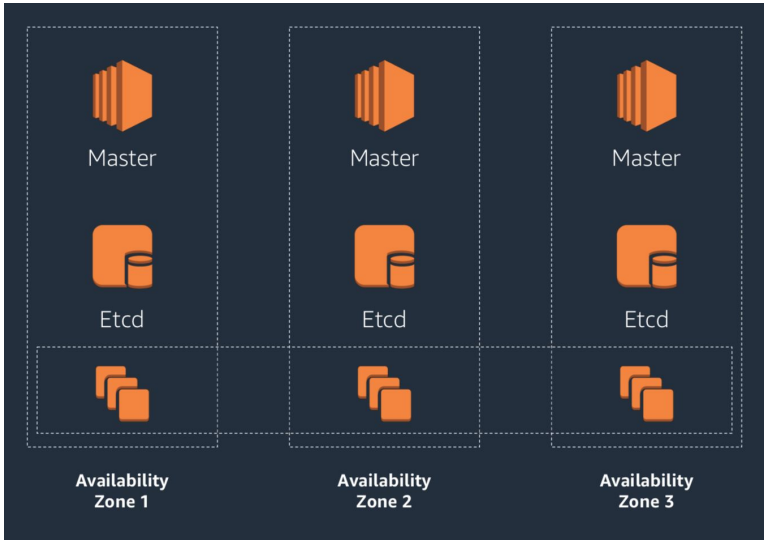
Kubernetes @ AWS

Mainly deployments happen via kops, kubeadm or templates like heptio-quickstart

Amazon contribute at the K8s AWS Special Interest Group

Over 62% of K8s workload runs on AWS

What is AWS EKS?



Master Nodes and etcd are controlled and managed by AWS

AWS ensure that there is always one node per Availability Zone running

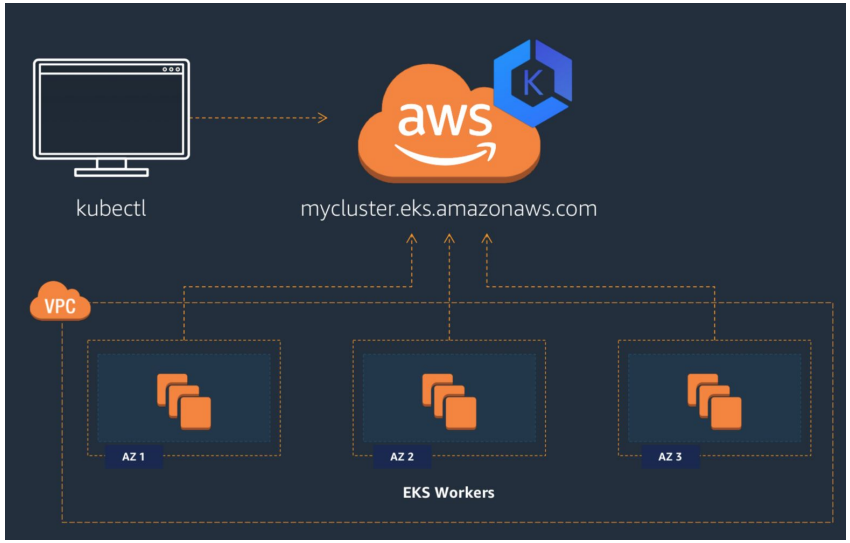
The worker nodes are up to your responsibility!

source: https://docs.aws.amazon.com/eks/latest/

AWS EKS Endpoint

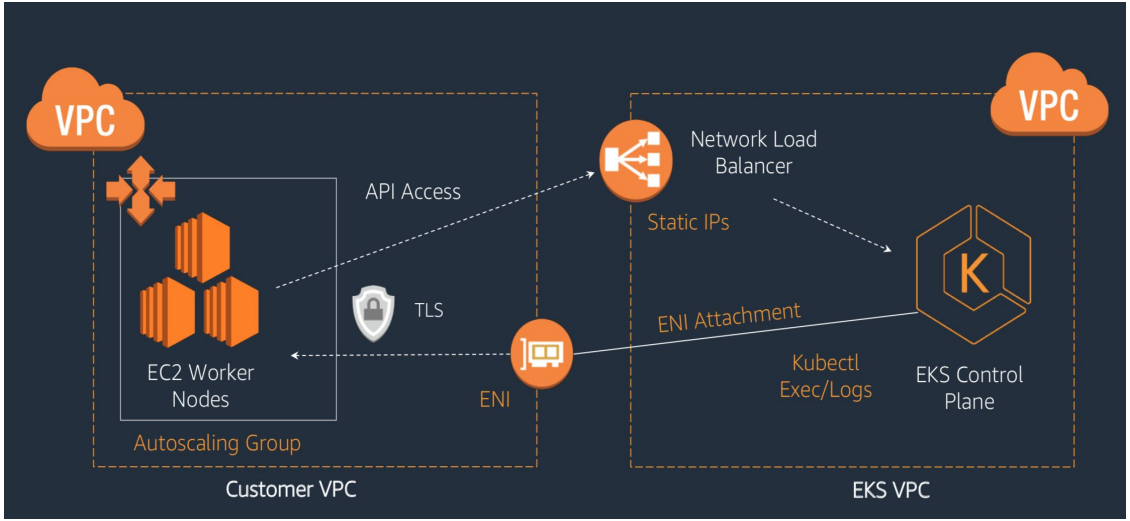
EKS publish your endpoint which you can reach by CLI/CI-Tool

This means you can use as usual the kubectl to control and manage your cluster



source: https://docs.aws.amazon.com/eks/latest/

How EKS CP talk to your worker



The EKS Control Plane and your worker run in different VPCs

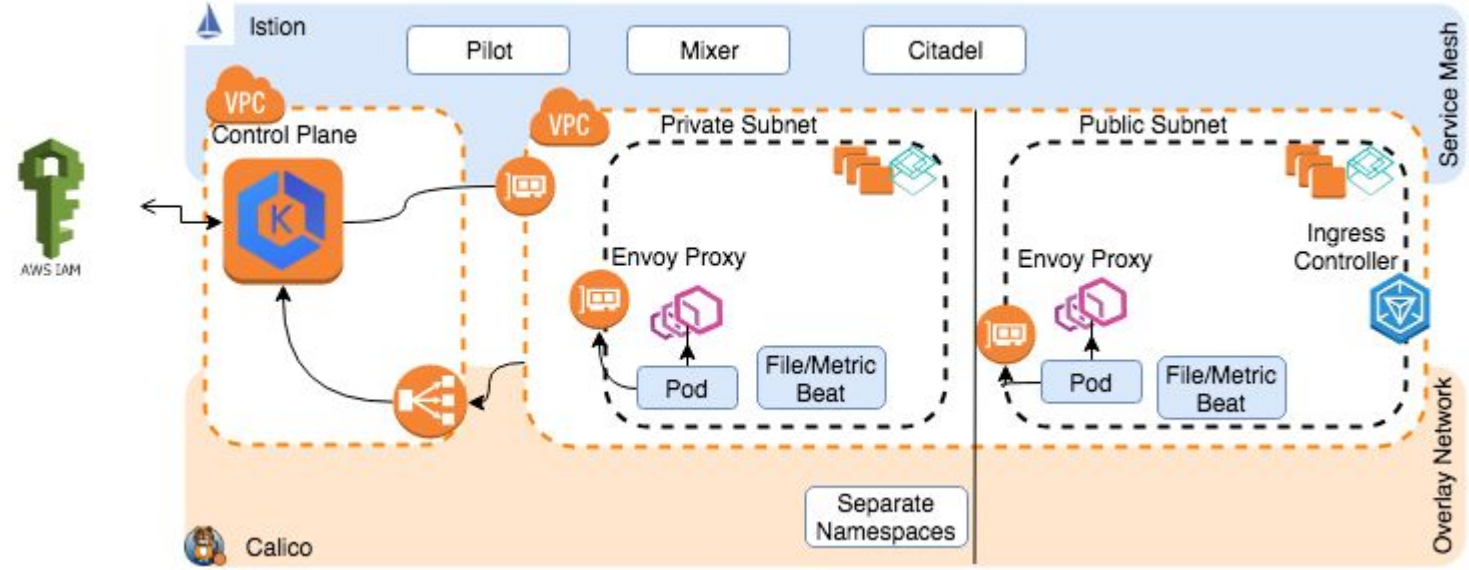
An ENI in you VPC is attached to the CP

While a Load Balancer sits in front of the CP

source: https://docs.aws.amazon.com/eks/latest/

What do you need for a production ready cluster?

A good basis for getting ready



Create the infrastructure

Cloud level

AWS managed VPC with 3 K8s master, one per each availability zone

One Auto Scaling Group for public and one ASG for private subnets

private & public subnets will be created per AZ (cannot span over multiple AZs)

VPC will span over 3 AZ in EU-WEST-1 (Ireland) region

Create the base infrastructure

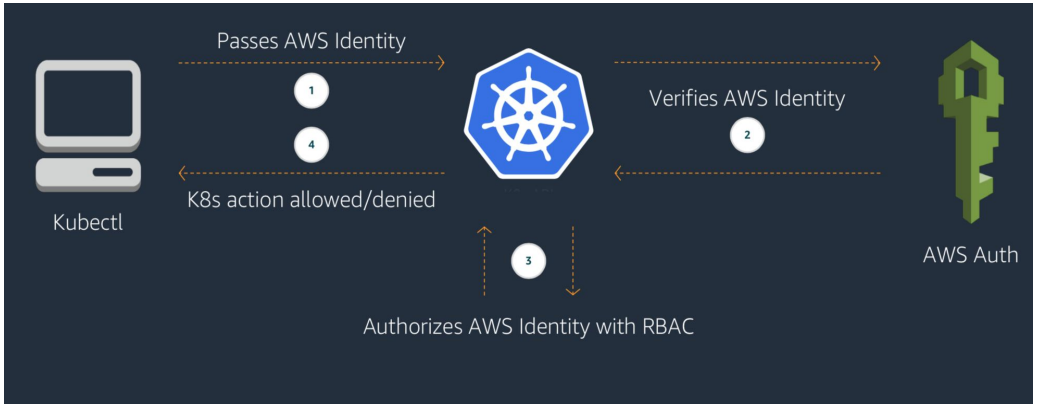
We need to create the EKS, a VPC for the worker as well as some subnets, security groups and auto scaling groups

A Terraform template makes this easy

```
→  terraform git:(master) ✗ terraform plan
→  terraform git:(master) ✗ terraform apply
```

Authorization & Authentication

IAM authenticator plugin



IAM manages the authentication

RBAC the authorization

After proving your identity you can use the K8s Endpoint as normal

source: https://docs.aws.amazon.com/eks/latest/

Deploy IAM Auth Plugins

1. Create IAM roles which will be assumed later
2. Specify the configuration map & demon set
3. Tell your API server to talk the auth server plugin
4. Adjust the K8s config:

```
# [...]
users:
- name: kubernetes-admin
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      command: aws-iam-authenticator
      args:
        - "token"
        - "-i"
        - "CLUSTER_ID"
        - "-r"
        - "ROLE_ARN"
  # no client certificate/key needed here!
```

```
- userARN: arn:aws:iam::000000000000:user/Alice
  username: alice
  groups:
  - system:masters
```

IAM Auth Plugin:
https://github.com/kubernetes-sigs/aws-iam-authenticator

AWS EKS CNI Networking

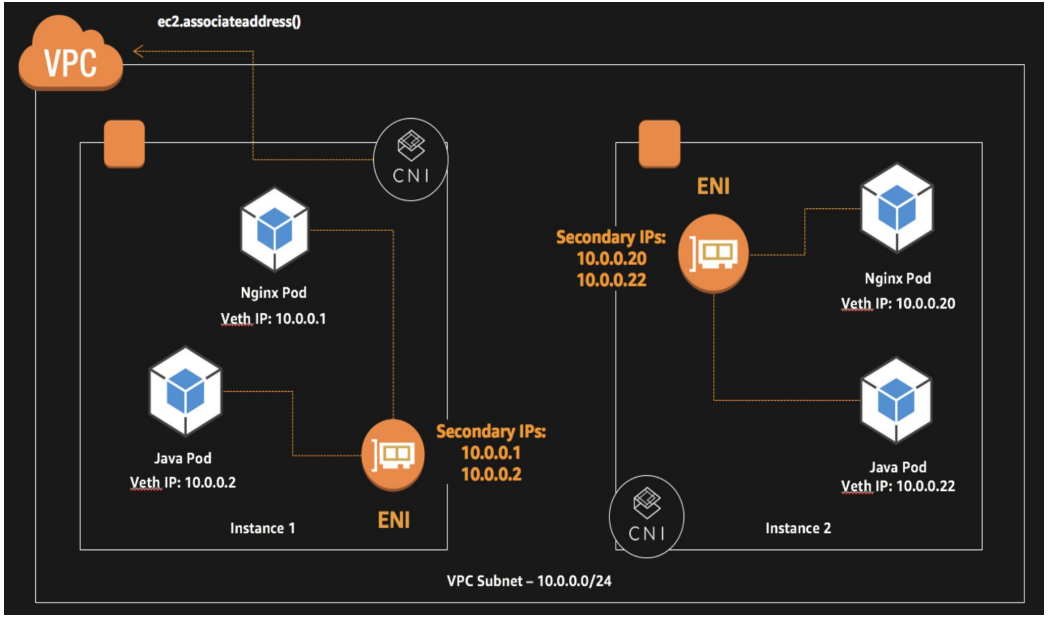
VPC native networking through CNI plugin

You can deploy a CNI plugin which bridges the gap between VPC and K8s

Each pod will get an IP

The maximum amount of pods per node depend on the node size e.g. m5.large

can have max. 3 ENI, each with 10 IPv4 addresses



source: https://docs.aws.amazon.com/eks/latest/

Seamless CNI integration

The CNI plugin is easy to setup:

```
→ kubectl apply -f aws-k8s-cni.yaml
```

Second, the long running node-Local IP Address Management (IPAM) needs a IAM role allowing the following:

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:AttachNetworkInterface",
        "ec2:DeleteNetworkInterface",
        "ec2:DetachNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeInstances",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:AssignPrivateIpAddresses"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:network-interface/*"
}
```

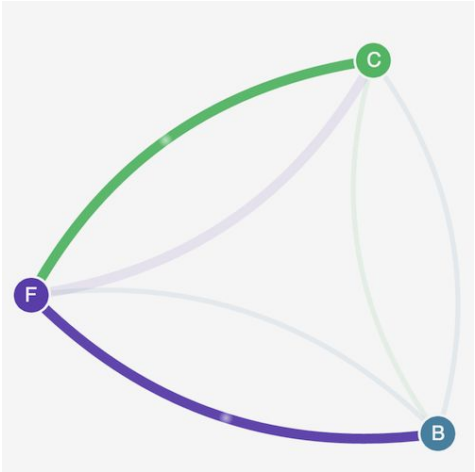
CNI Plugin:
https://github.com/aws/amazon-vpc-cni-k8s

Implement the Overlay Network

Calico

The overlay network help you to secure and isolate the namespaces on cluster level

Therefore Calico can block
or allow dedicated communication
paths between namespaces
and pods




```
→ kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/config/v1.2/calico.yaml
```

Managing the overlay network

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

```
→ kubectl apply -f ./my-deny.yaml
```

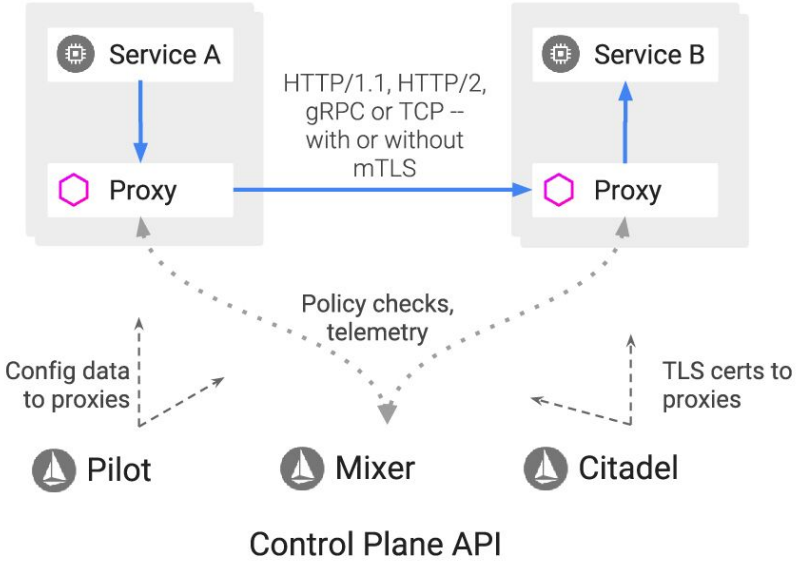
```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
          port: 80
```

Service Mesh

The service mesh secures the communication between services allows layer 7 routing

Normally a sidecar injection deploy a proxy to each pod

It brings also basic ingress controller



Deploy your Service Mesh

A default deployment with auto. sidecar injection looks like this

However you still will need to care about security

```
helm install \
--wait \
--name istio \
--namespace istio-system \
install/kubernetes/helm/istio

kubectl label namespace default istio-injection=enabled
```

Monitoring & Logging

Finally you need some monitoring & logging

```
→ kubectl create -f filebeat-kubernetes.yaml
→ kubectl create -f metric-kubernetes.yaml
```

Therefore you can use a elasticsearch on AWS as service endpoint for your beats or fluentd

Configure the yaml and here you go

Takeaways

Keep clusters simple: Complexity doesn't bring security, it just increase your effort

First learn, then optimize: Do not try to predict the workload, observe it and adjust the instance types

Utilize "as a Service" Backends: Many companies want to host their own DB or even run it on K8s; DBaaS are critical resources when you reach the point of data protection, availability and HA; also messaging and other resources can be helpful