Part of overall transformation initiatives over last few years, I have been advocating need to move to containers in organization. We started with docker container. I have setup few docker swarm clusters, enabled CI/CD to deploy to docker cluster but then as industry moved to more mature container orchestration platform, switched focus to Kunbernetes.

Last year , I was involved in project where we wanted to move footprint we had in docker swarm as well as number of COTS application to K8 cluster. I have evaluated different k8 offering….Had POC to build on prem K8 cluster as well in AWS EC2 machines. Looked into AWS ECS & EKS.

Requirements we had is this k8 cluster should be scalable, secure, highly available & less management overhead for day 2 operation.

Given amount of work involved to build & manage K8 onprem cluster or on EC2…I have finally proposed to make use of AWS EKS cluster & worked to create overall EKS architecture & integrate that with rest of the tooling so it can be used to migrate applications.

AWS EKS had also 2 types…one is self-managed/AWS managed nodes ..& other with Fargate. For now, we went ahead with EKS with managed instance. With AWS managed node group - there is no pod disruption & AMI are managed by AWS. ( custom AMI not allowed ) .

Best thing is it uses Autoscaling group  under the hood so no application downtime during upgrade as it maintain min number of EC2 in managed group.

EC2 worker node - whct instance to use was driven by how many pods you need as each instance has specific limit.

Since control plane is managed by AWS in case of EKS< I didn't have to size or think about HA for k8 controller, API server or etcd database. With EKS, we get highly available , multi zone control plane so all we need to do is make sure our worker nodes are spread out in multiple zones. There were still number of other things which I looked into to make sure the cluster pattern I build is repeatable pattern so anyone who need EKS cluster, they can provision it with same standards.

* AWS control plane -
  + AWS takes care of scaling , maintain, upgrade of control plane components. Maintain high availability of EC2 in multiple AZ
  + Master & etcd runnign in 3 AZs

* CNI plugin - AWS VPC  - reason to use this plugin over NSXt or weave  or other overlay plugin is native integration of this with AWS services.
* scalability -  Used HPA & cluster autoscaler
* Overprovisoing EC2
  + EKS cluster comes pre-configured with an overprovisioning component that reserves compute resources so pending pods can be immediately scheduled and serviced without waiting for new EC2s to get deployed. This overcommitment allows the cluster to respond to demand quickly without getting bogged down by intervening processes such as firstboot.
  + So used - [cluster-proportional-autoscaler](https://github.com/kubernetes-incubator/cluster-proportional-autoscaler)to have that buffer capacity build …so in case of 10 node cluster with 4 cores each & if we needed 20% buffer..than that’s 2 nodes extra.

* To integrate k8 with onprem AD-ENT - Kubernetes identity - uses Dex OIDC k8 plugin as OIDC provider.
* Integration with ARGO CD tool -
* Storage plugin - Dynamic provisioing - EBS & EFS
* EKS security - along with layered with other security to make this
  + Used POD level security group to control traffic in & out of k8.
  + Proposed Namespace isolation
  + Recommended use of  Sealed Secret - Uses CRD & then use private/public key pair to seal secrets.
  + Sysdig
* Following sysdig agent would be deployed as daemonset on each protected worker node.
  + sysdig-agent
  + sysdig-image-analyzer
* OPA gatekeeper agent -
  + All images sourced from approved artifactory
  + Prevent deletion of sysdig/twistlock daemon
* PodSecurityPolicy

securityContext:

  allowPrivilegeEscalation: false

  readOnlyRootFilesystem: true

  runAsNonRoot: true

  runAsUser: 1000

* Ingress controller - AWS ingress ALB.
* CoreDNS
* Used fluetd agenst & kinesis firehose to push logs to onprem ELK/ SIEM solution
  + logging through fluetd ..& then pushing logs to prometheous …& then visualizing through graphana dashboard
  + one colud use cloudwatch container insight by sending logs to cloudwatch from EKS cluster

I have completed this implementation & documented it. Worked with automation team to  create jenkins jobs & cloudformation template to provision this EKS cluster with all controls, drivers, plugins..so anyone who need new EKS cluster, iniaitiate jenkins job, passes specific paramters…& behind the scene cloudformation build EKS cluster.

Few additional thing which I have been trying to get done is to get confluent kafka build in K8 using confluent operator so that not only day 1 but day 2 operation to stand up & manage complex infrastructure needed for Kafka or Mongo is ready & build on highly scalable, abailable & reliable underlying k8 cluster.