PRODUCTION ENVIRONMENT WITH EKS AND ALB

AWS EKS managed node groups or self managed ec2 instances

I'm happy to use either of the methods to provision eks clusters in production rather than fargate method (Fully managed by aws and no control over nodes), By using managed node groups, EKS cluster itself will provision the nodes with autoscaling groups.

Cluster autoscaler

By default EKS cluster provisioned with node groups does not scale the node group even though nodes are added to the autoscaling group as there are no scaling policies added.. Therefore , we need to install cluster autoscaler and it will scale the kubernetes cluster accordingly.

With managed node groups,

- Easy provisioning
- Can use different ec2 instance types, AMI, user data (now it is supported) and subnets(azones) based on our requirements.
- AWS manages the node version upgrading, graceful node drain

With self managed node groups,

- More control over node resources
- needs to create Auto Scaling groups and launch configuration/launch templates
- Need to manage node registration with the eks cluster by having a userdata

For instance :

```
/etc/eks/bootstrap.sh ${cluster_name} ${bootstraparguments}
```

AWS VPC Architecture

Subnets

I recommend to use both Public and private subnets when we provision eks cluster

Public subnet	**Private subnet**
To create ALB in public subnets	To provision eks node groups

SInce we use both public and private subnets , we need to create additional VPC resources in aws for access and manage internet access to private subnets.

- A NAT gateway in one of the public subnets
- A route table association with private subnets with NAT gateway

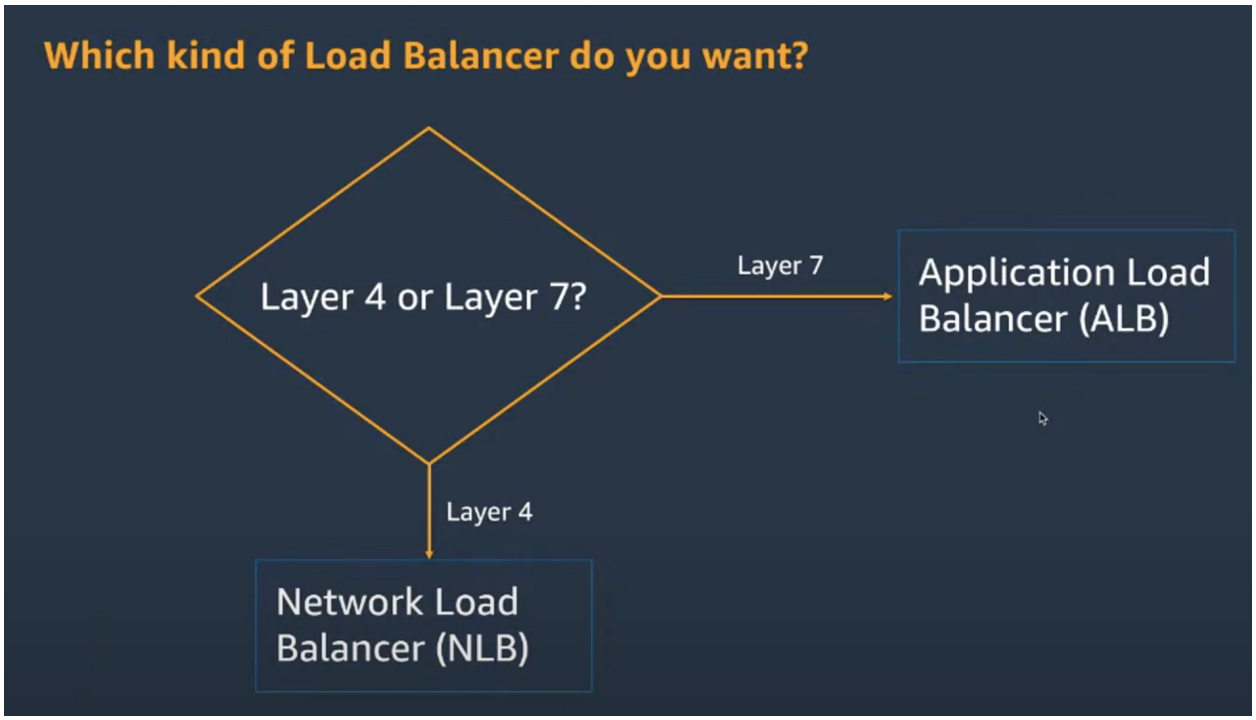
AWS Load Balancer ingress controller with ALB (application load balancer)

The ALB Load Balancer ingress controller is specifically designed for aws cloud LB provisioning and has full support by AWS community.

With AWS Load Balancer controller ;

- We can choose ALB or NLB
- The ALB Ingress controller creates ALB for every ingress group mentioned in the ingress Objects.
- We can have a single ALB for one cluster or can divide based on application grouping(ingress groups functionality in AWS LB controller)

Application Load Balancer (ALB)



Based on our requirements, we can choose the LB on aws.
If ,

1. We have HTTP web applications or web services
2. LB to handle redirect
3. Direct inbound traffic to different target groups

Then, ALB is the right choice, on the other hand, if we have TCP/UDP traffic or have a non-http application or need end to end encryption, then NLB is the right choice.

Helm charts

Since we use micro service architecture on kubernetes, helm is the best extension or encapsulation that we can use to deploy micro services. Helm is considered as the package manager for kubernetes.

1. Easy to maintain with different approaches (central helm chart, umbrella, individual chart)
2. Reusable go chart templates

3. Large open source community and help
4. Great for application and chart versioning

Monitoring && Logging

We have multiple choices for a monitoring tool from open source to paid/licensed managed cloud tools.

I would prefer to go with ***Datadog*** as the monitoring tool in the production environment. DataDog is a managed Monitoring solution where we get all the aspects of a monitoring tool including alerting, integration with different tools and APM.

- Daemonset deployment
- Easy integration with cloud providers
- Easy integration with communication channels like slack, hangouts.
- The perfect option for logging as we can control the logging functionality in DD deployment, can enable logs for particular micro services or all services depending on the requirements.
- Easy integration with pagerduty and get notified in on call.
- Options for synthetic checks (an alerting system for internal facing backend applications)
- We can create different monitors and alerts
- We can monitor databases in cloud
- We can implement APM solution in DD
- ***Good Documentation and customer support***

Hashicorp Terraform

Infrastructure Provisioning and management

I would prefer to use Terraform to provision infrastructure on cloud.

- While we manage infrastructure with terraform, the team get to know the state of the particular resources and store the state in one central location (**preferably AWS s3**) .

- I also prefer to use **DynamoDB(Best use case when we work in a team)** to store the Lockkey, so that everyone in the team get to know if someone is working on a particular project (terraform apply).

- Terraform is modular and we can create one module and use for different environments

- Large open source community and **documentation**

- Integration or provider plugins available for all major cloud providers

- **Disaster Recovery**
  With a production ready TF code, we can provision the infrastructure in another region of aws in the case of a region failure.

Hashicorp Vault and Vault Injector

Secret management in Kubernetes

- Hashicorp vault is one of the best methods to store secrets in the kubernetes environment along with vault injector.

- It gives the functionality to export secrets during the pods startup and no need to maintain kubernetes secrets on the cluster that gives extra security

Kubernetes Application Deployment (Helm) - CI/CD

Tools for the deployment process : jenkins
Source code management : Github
Docker container registry : ECR on AWS
Kubernetes package management : Helm

With jenkins as the CI/CD tool, we can use Jenkinsfile deployment pipeline stored along with the source code of the application.

We can write two types of jenkins pipeline files

1. Declarative
2. Scripted

I used to deploy applications with scripted pipelines before which is easier to integrate with bash and python along with groovy, on the other hand I like the Declarative method more now as it has more syntax and makes the pipeline code easier to read and write.

Container Image store - ECR

One of the crucial aspects of the ECR is that IAM handles authentication and authorization for the container registry. Therefore, It is easy to access ECR from all the different services AWS provides -> ECS, EKS and many more.

Source code Management - Github

Source code management is crucial in any development practices and github is the public cloud platform most companies use where,

- we can create an organization and add/manage users
- Manage groups and access policies
- Create different branch protection policies
- Easy Integration with different CI/CD tools- azure devops, jenkins webhooks, gitops
- Easy integration with sonar cloud for code scan

Code scan - SonarCloud

- Sonar cloud is the cloud platform provided by sonarsource company which allows us to integrate our github repos to sonarcloud and scan whenever we create a pull request to different branches and more capabilities.

- We can add sonar scan as one of the steps in the CICd pipeline

Access Management - AWS IAM Roles

Security is the most promising thing in aws when we compare with other clouds, the flexibility of assigning different policies and creating inline policy for the better level of security.

1. Give least permissions always whenever possible.

2. Use IAM roles over hardcoring aws access creds of a user.

3. Create inline policies so that we give access to a particular aws resources only

DNS Management - AWS Route53

Route 53 is the ideal solution for the cloud dns management for eks. The service will connect incoming requests to Load balancers, ec2 instances or s3 buckets easily.

- Has Health check options
- Different routing mechanisms
- Domain name registration

Content Management - AWS Cloudfront

AWS cloudfront is ideal for content management for applications hosted in aws like ec2, eks, s3 and more.

- It eliminates latency issues and better user experience
- Different data source types/origins (s3, LB, instances with http web server)
- Option for Invalidating a file
