**PLEASE DELETE THIS PAGE. A COPY HAS BEEN CREATED IN SHARED FOLDER  
  
  
  
  
K8s cluster common security**

* Should have different namespaces for different environments.
* Firewall rules should be placed on all nodes of the k8s cluster with required ports only.
* Make sure K8s all nodes should not have security vulnerability.
* root access should be disabled on every node within the cluster.
* K8s cluster, docker images, OS , and running software all need to be updated with latest stable updates/patches.

**API server**

* An API server which is a central interface needs to be strictly implemented by authentication, Authorization and access control which should be integrated with organization access control such as active directory, LDAP, and SAML.
* The insecure port should be disabled in API server yml “/etc/kubernetes/manifests/kube-apiserver.yaml” with [--insecure-port=0](https://kubernetes.io/docs/reference/access-authn-authz/controlling-access/#api-server-ports-and-ips)
* Implement [audit logger](https://kubernetes.io/docs/tasks/debug-application-cluster/audit/) to record API server events.

**Authentication**

* [X.509 certificates](https://kubernetes.io/docs/concepts/cluster-administration/certificates/) should be implemented.
* Create a token with [OpenID Connect](https://kubernetes.io/docs/reference/access-authn-authz/authentication/#openid-connect-tokens) (JSON Web Token-This token is a JSON Web Token (JWT) with well-known fields, such as a user’s email, signed by the server.)
* Don’t use the default [Service Account](https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/).
* Credentials should be rotated after a particular time.

**Authorization**

* [Node authorization](https://kubernetes.io/docs/reference/access-authn-authz/node/)(It grants permission to kubelets based on the pods they are scheduled to run)
* [Attribute-based access control](https://kubernetes.io/docs/reference/access-authn-authz/abac/)  (It provides access rights to users through the use of policies)
* [Webhook](https://kubernetes.io/docs/reference/access-authn-authz/webhook/) should be placed for authorization.
* [Role-based access control (RBAC)](https://kubernetes.io/docs/reference/access-authn-authz/rbac/)(It is a method of regulating access to a computer or network resources based on the roles of individual users within an enterprise)

**ETCD**

* TLS [certificates](https://kubernetes.io/docs/setup/best-practices/certificates/) should be enabled to communicate only with api-server.
* “--auto-tls=false” no to self-signed cert.
* “--trusted-ca-file” Only Certificate Authority signed certificates should be used.
* Etcd backup should be encrypted with full disk encryption.

**Secure Secrets**

* Storing Secrets in ECTD should have to use [encryption methods](https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/)

(AES-CBC → strong, fastest)

* Secrets can also be encrypted with Third-Party Stores ([HashiCorp Vault](https://blog.aquasec.com/managing-kubernetes-secrets#:~:text=By%20default%2C%20data%20in%20Kubernetes,save%20them%20as%20environment%20variables.))
* Don’t Build Images with secrets (it may ask to rebuilt and redeployed whenever secrets change which is not a ideal way)
* Don’t pass secrets as Environment Variables(this can be visible through inspecting containers)
* Try pass Secrets in Files( Provide abstraction layer which is clearly safe)
* Secret Rotation and Revocation (Changing secret in cycle basis and removing them when they have no further use.)

**Kubelet**

* Kubelet should have anonymous requests to false [--anonymous-auth](https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/#options)=false
* read-only port for the Kubelet to serve on with no authentication/authorization [--read-only-port](https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/#options)=0
* The TLS [certificate](https://kubernetes.io/docs/setup/best-practices/certificates/#certificate-paths) should be enabled to communicate with api-server.

**Kubernetes Dashboard**

* It should be accessible by the private network/IP.
* Make sure the Dashboard service account has limited access([RBAC enabled](https://medium.com/@int128/access-kubernetes-dashboard-with-rbac-f175e2cf3c01))

**Securing the Image**

These points need to be implemented to secure images.

* Image Trust and Supply Chain([grafeas](https://kubernetes.io/blog/2017/11/securing-software-supply-chain-grafeas/)) : this inspects images for critical data.
* [Minimizes Images](https://blog.logrocket.com/reduce-docker-image-sizes-using-multi-stage-builds/) to reduce surface attack.
* Configure a [private registry](https://docs.docker.com/docker-hub/repos/#private-repositories) with third party like GitLab’s Container Registry, jrog, nexus.
* AlwaysPullImages=true(Pull the image from private registry every time whenever needed)

**Running container Securely**

* Use the least privilege to carry out the task at hand.
* Do only the minimal host mounts if required.
* Limit communication between pods by implementing ([Network Policy](https://kubernetes.io/docs/concepts/services-networking/network-policies/)).
* Say no to the root.
* Avoid unnecessary package installation.
* pod should have [security context](https://kubernetes.io/docs/tasks/configure-pod-container/security-context/) implemented to control container privileges.
* Prevent container to load unwanted kernel modules by creating rules to block in file /etc/modprobe.d/kubernetes-blacklist.conf
* Also enable selinux (module\_request) to completely deny loading any kernel module.
* Also need to implement network policy to pods to access with cloud metadata API.