Securing Kubernetes: Best practices and effective strategies

Nilesh Jayanandana, Architect at Insighture

Kubernetes Attack Surface

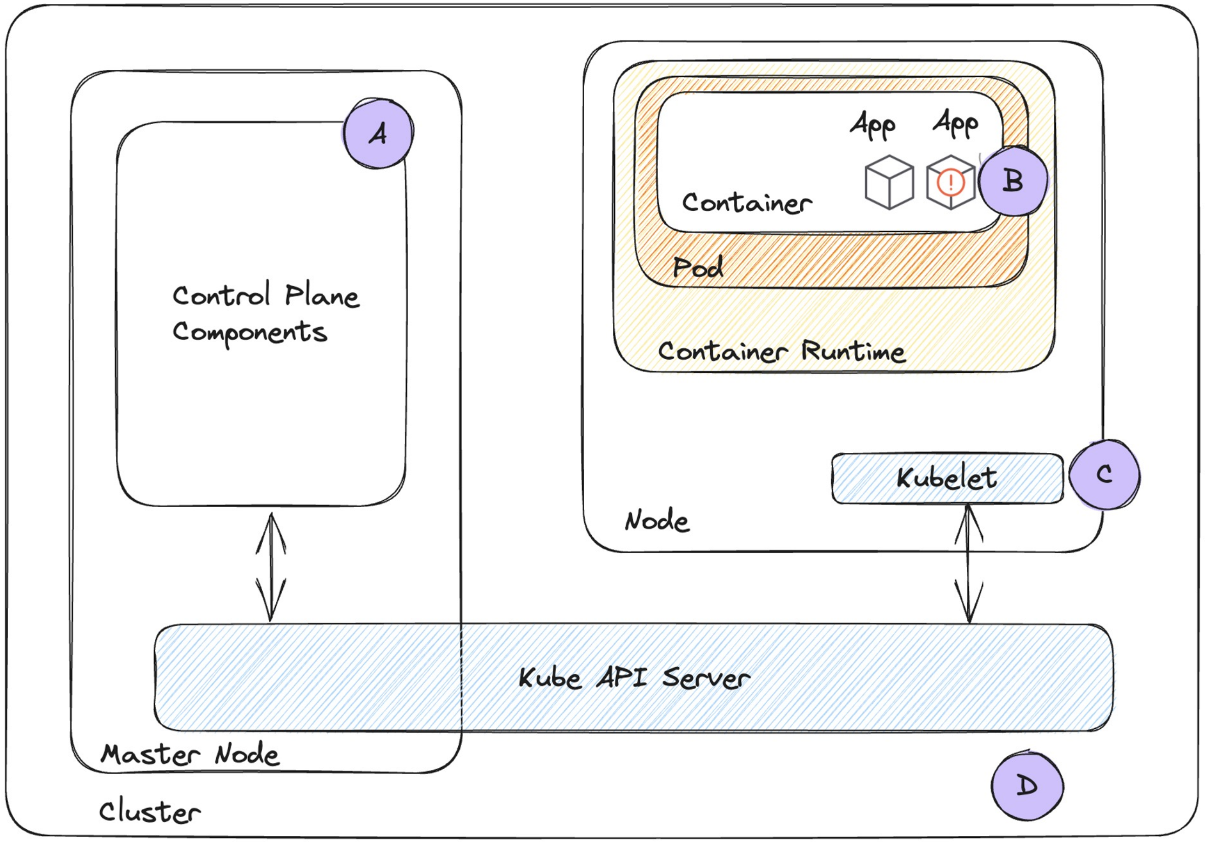
A Access via Kubernetes API Proxy, etcd API

B Exploit vulnerability in apps or 3rd party libraries

C

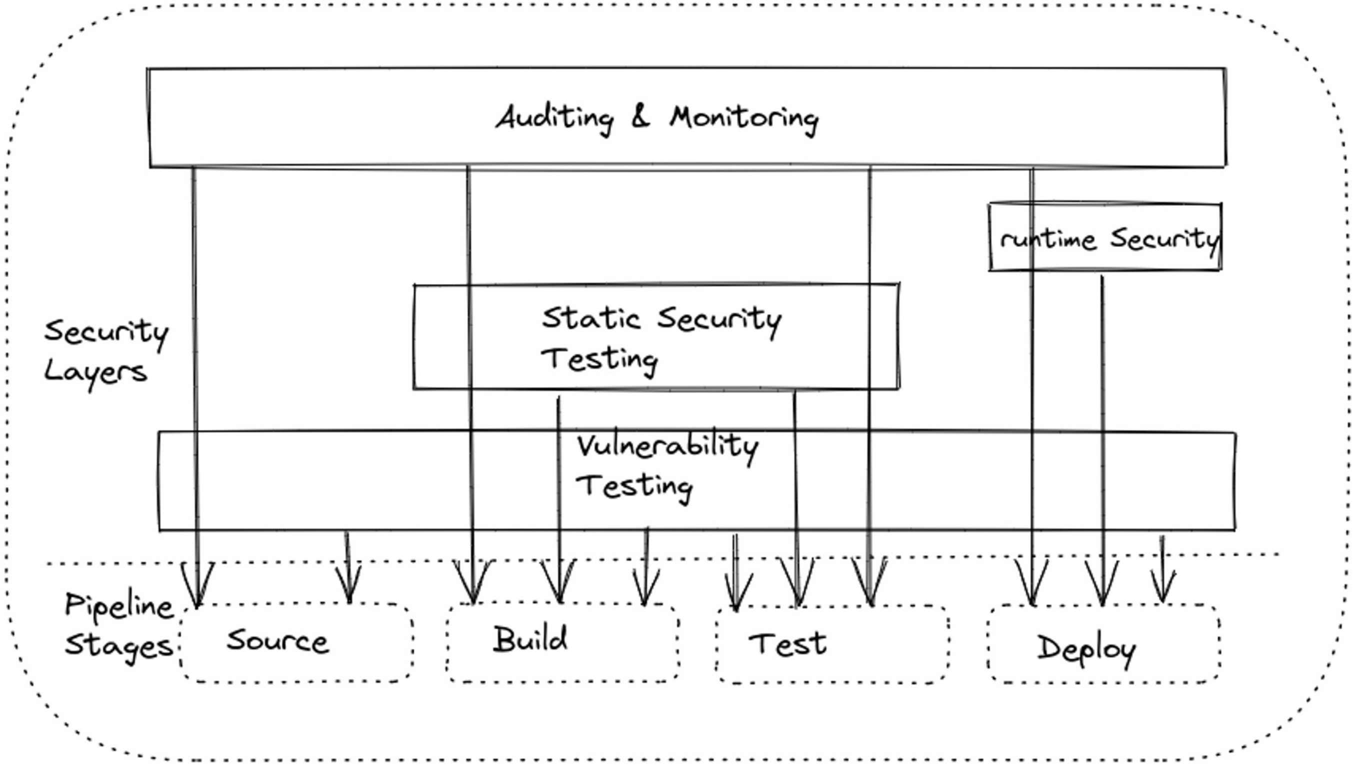
D Access via Kubelet API

Access to the



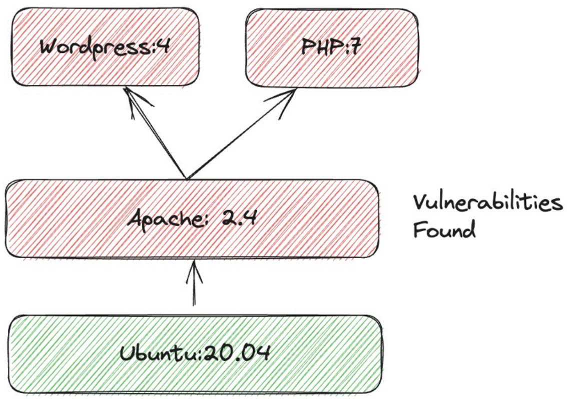
Protect your Application Runtime

Your CICD Pipeline



Static Security Testing

- Image Vulnerability Scanning
  - Web Servers or other apps can contain vulnerabilities.
  - Exploiting may lead to
    - Privilege escalation
    - Remote Shell access
    - Information Leaks
    - DDOS
  - Use tools like Clair, Trivy or any tool to scan your image.

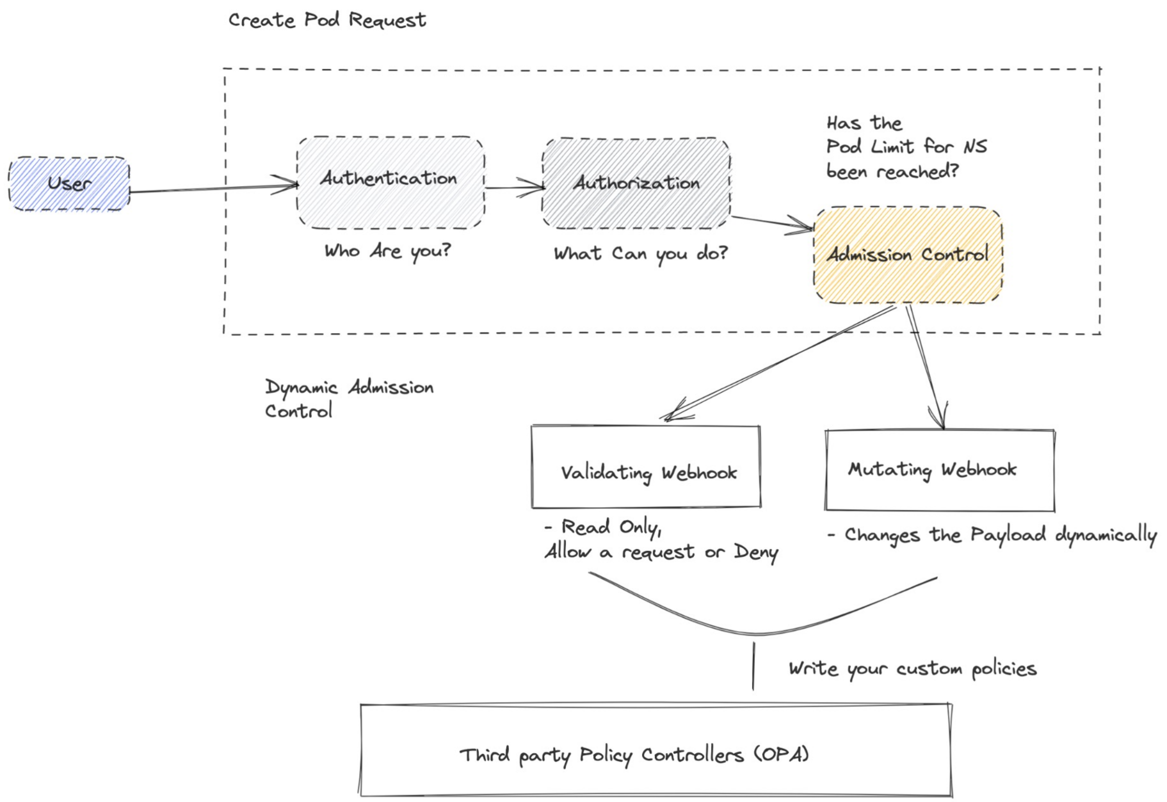


Static Security Testing

- Code Vulnerability Scanning
  - Make sure your libraries and dependencies are upto date and not running any version of vulnerable dependencies. (Eg: Log4J)
  - Make sure you have not hardcoded any sensitive information into the code.

- Configuration Scanning
  - Use tools like checkov, Conftest, KubeSec to enforce security standards in your YAMLs

Kubernetes Admission COntroller



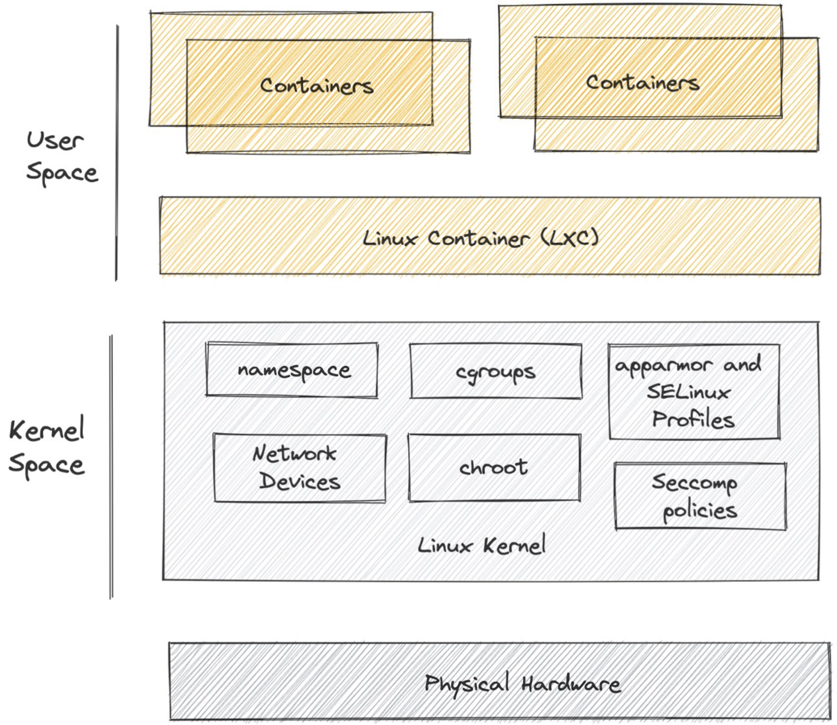
Container Hardening

- Make sure your containers are immutable
  - Remove Bash/Shell
  - Make file system Read Only
  - Run as user a non root user

- Enforce these on Kubernetes level without changing the image
  - Use Startup Probe to remove bash
  - Set RunAsGroup, RunAsUser and RunAsNonRoot with Security Contexts
  - Enforce these rules with Open Policy Agent and GateKeeper.

Container Runtime Security

- Disable privileged containers and privilege escalation
- Drop all capabilities and add only the ones that are needed
- Use AppArmor or seccomp profiles to restrict processes running in Containers.
- Container Sandboxing
- Use a tool like Sysdig falco to monitor abnormalities in container runtimes.



Conceal Information

- Don’t use environment variables to inject sensitive information to your containers
    - Inject via a Secret Manager
        - Hashicorp Vault, Azure KV, AWS Secret Manager, Google Secret Manager
    - Mount Secrets as Files

- Make your container root file system read only

- Do not log sensitive information - Developer discipline

Isolate your tenants on the Network

- Use Namespaces

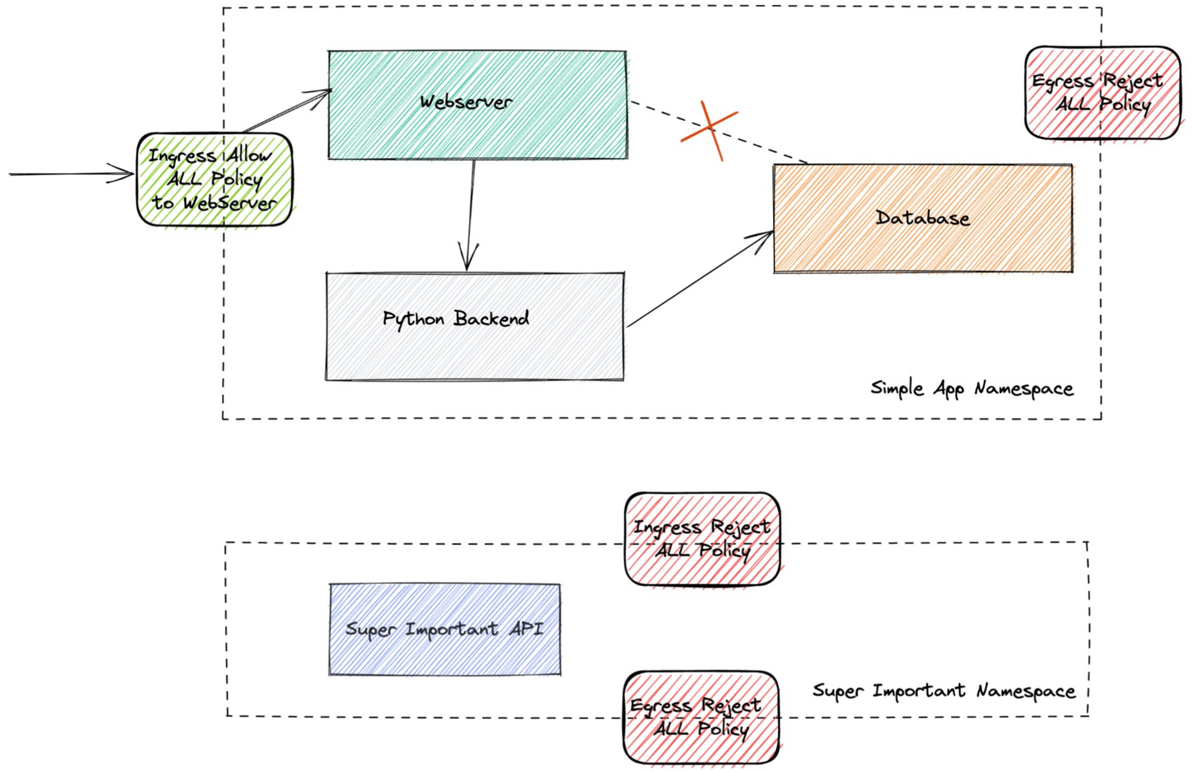
- Use Network Policies - Kubernetes Equivalent for Firewalls
    - Disable access between namespaces with network policies with a deny all rule.
    - Selectively allow ingress and egress rules as and when needed for your tenants.
    - Make sure your CNI supports Network Policies in K8s

- Make even inter service communication go through an API Gateway

- Use mTLS easily with support of Service Mesh or an ebpf enabled CNI

- Use admission controller to enforce policies on the cluster.
    - Open Policy Agent
    - Kyverno

Network Policies

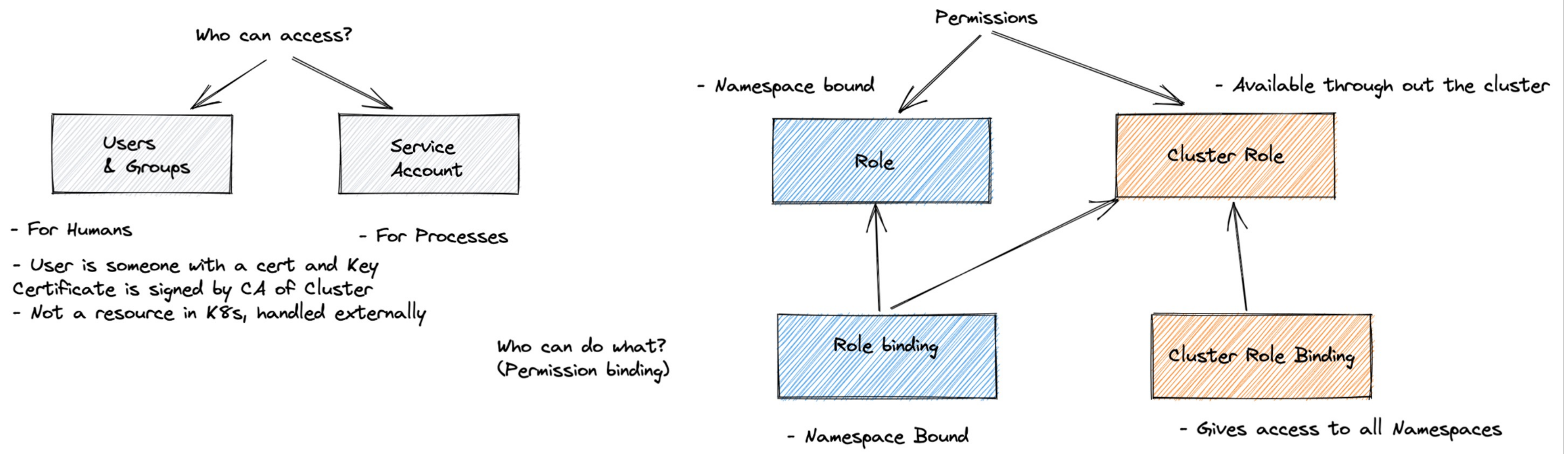


Protect your Control Plane

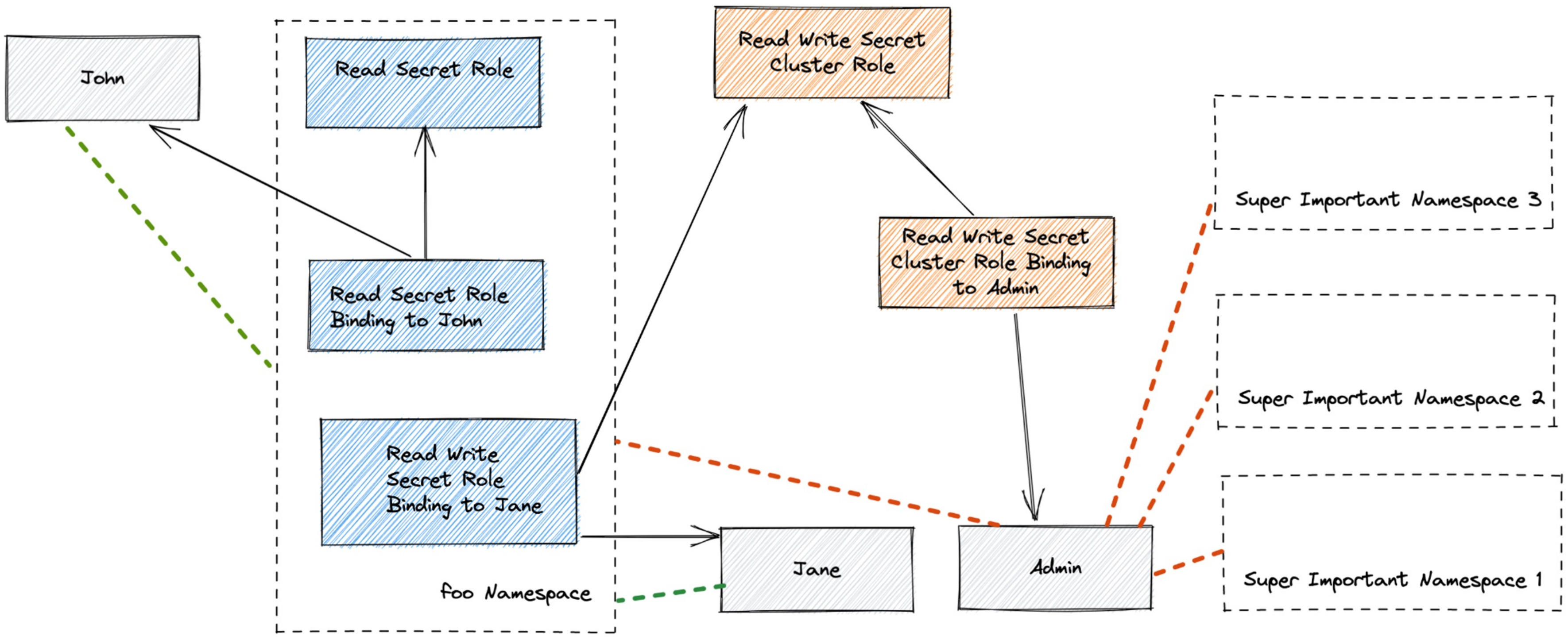
Hardening your Kubernetes Cluster

- Have proper RBAC policies set in your cluster
  - Create Accounts , Certificate Signing and Role Binding
  - Avoid cluster role bindings
- Enable Audit Logging
- Run CIS Benchmark and do recommended fixes
  - Kube Bench
- Use CIS Hardened Node Images
- Encrypt ETCD
- Mount your Secrets as files and not as Environment Variables

Kubernetes RBAC



Kubernetes RBAC Example


John
Read Secret Role
Read Secret Role Binding to John
Read Write Secret Role Binding to Jane
foo Namespace
Read Write Secret cluster Role
Read Write Secret cluster Role Binding to Admin
Jane
Admin
Super Important Namespace 3
Super Important Namespace 2
Super Important Namespace 1

Final Thoughts

- Kubernetes security is still new and vulnerabilities get patched frequently.

- Lot of the cloud managed Kubernetes have the control plane security properly managed.

- Many organizations need some level of multi tenancy within their applications in K8s Clusters. These could be their internal apps or apps running by an outsourced vendor.

- If proper standards and best practices are followed and you keep upto date with latest releases and patches, things should be alright.

Thank You!