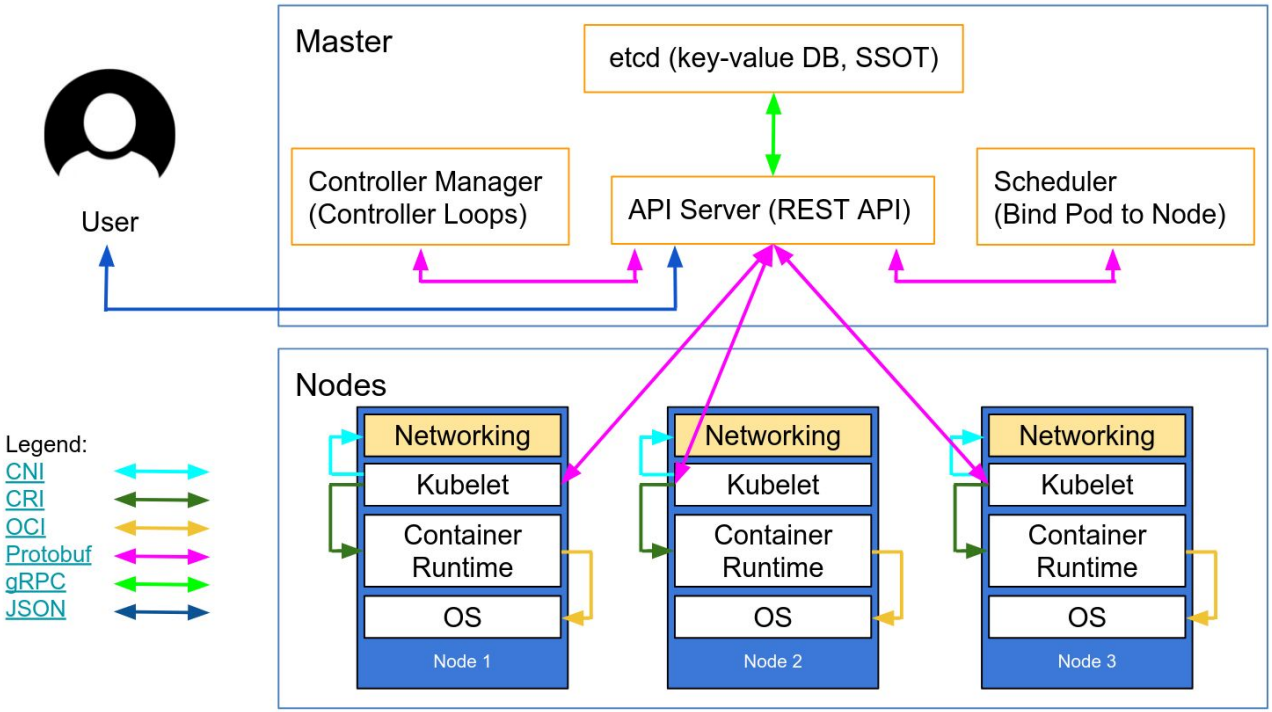
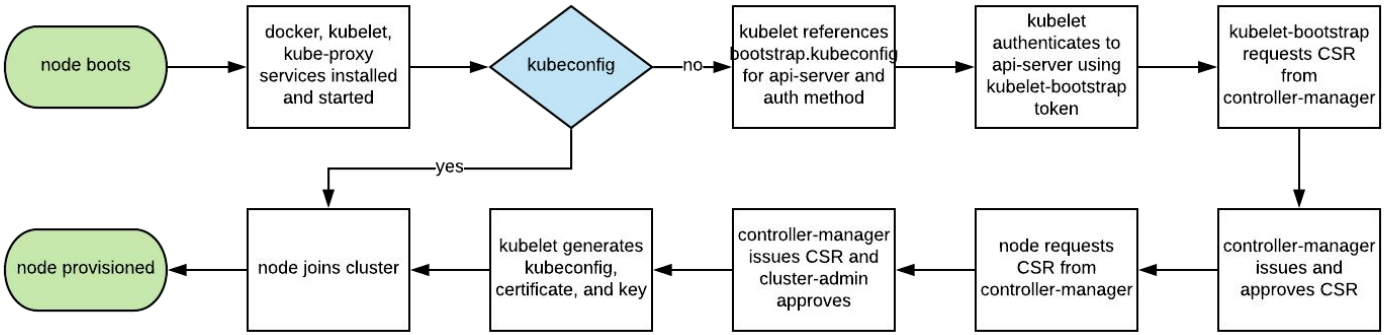
kubernetes

11 Ways (Not) To Get Hacked



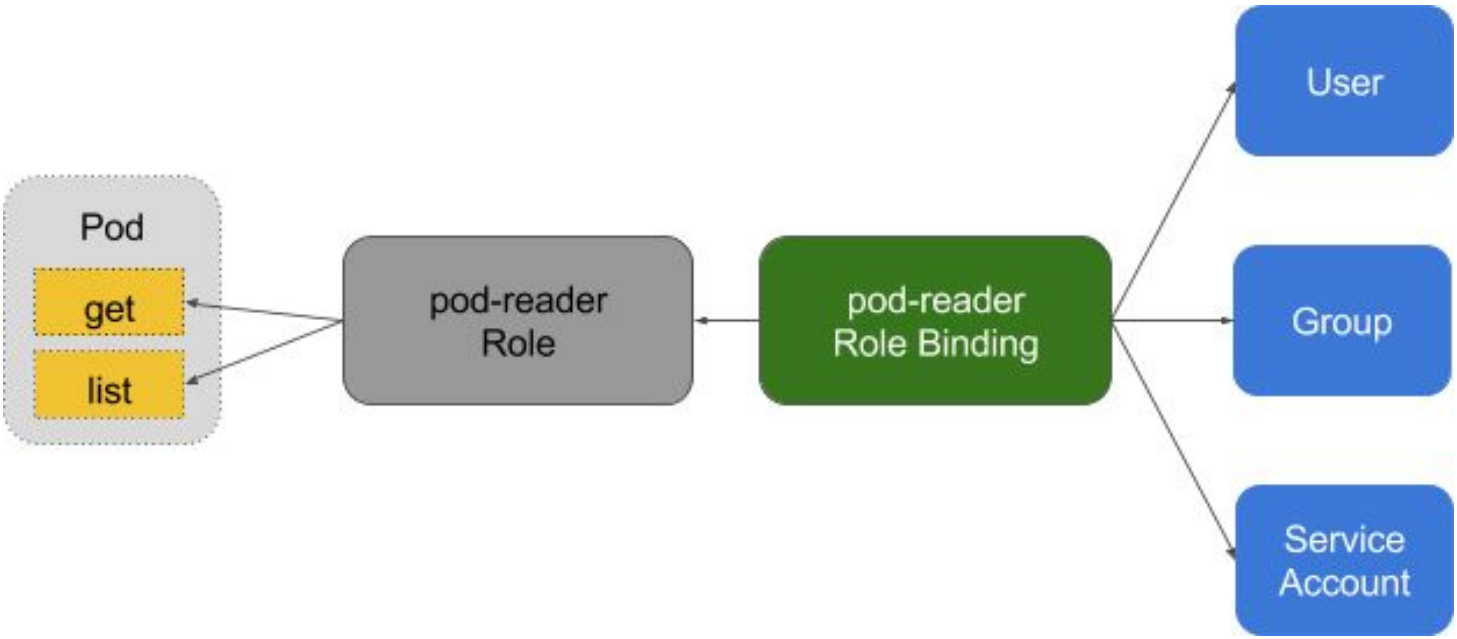
1. TLS Everywhere

TLS should be enabled for every component that supports it to prevent traffic sniffing, verify the identity of the server, and (for mutual TLS) **verify the identity of the client.**



Autoscaling Kubernetes nodes was historically difficult, as each node requires a TLS key to connect to the master, and baking secrets into base images is not good practice.

Kubelet TLS bootstrapping provides the ability for a new kubelet to create a certificate signing request so that certificates are generated at boot time.



2. Enable RBAC with Least Privilege, Disable ABAC, and Monitor Logs

Role-based access control provides fine-grained policy management for user access to resources, **such as access to namespaces.**

3. Use Third Party Auth for API Server

Centralising authentication and authorisation across an organisation (aka Single Sign On) **helps onboarding, offboarding, and consistent permissions for users.**

Integrating Kubernetes with third party auth providers uses the remote platform's identity guarantees and prevents administrators having to reconfigure the Kubernetes API server to add or remove users.

4. Separate and Firewall your etcd Cluster

etcd stores information on state and secrets, and is a critical Kubernetes component - it should be protected differently from the rest of your cluster.

etcd should be configured with peer and client TLS certificates, and deployed on dedicated nodes.

5. Rotate Encryption Keys

A security best practice is to regularly rotate encryption keys and certificates, in order to limit the "blast radius" of a key compromise.

Kubernetes will rotate some certificates automatically (notably, the kubelet client and server certs) **by creating new CSRs as its existing credentials expire.**

6. Use Linux Security Features and PodSecurityPolicies

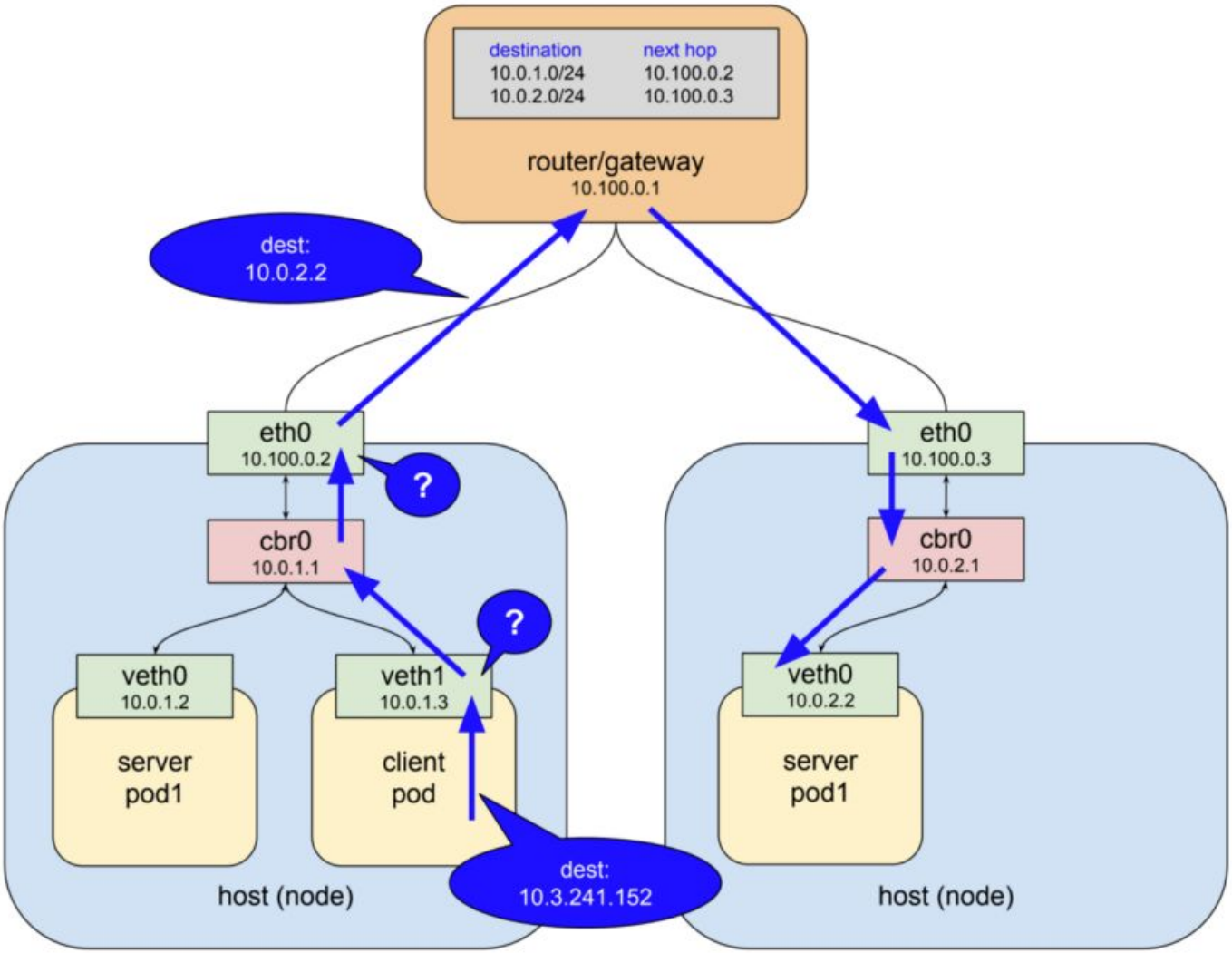
The Linux kernel has a number of overlapping security extensions (capabilities, SELinux, AppArmor, seccomp-bpf) that can be configured **to provide least privilege to applications.**

7. Statically Analyse YAML

Where PodSecurityPolicies deny access to the API server, static analysis can also be used in the development workflow to **model an organisation's compliance requirements or risk appetite.**

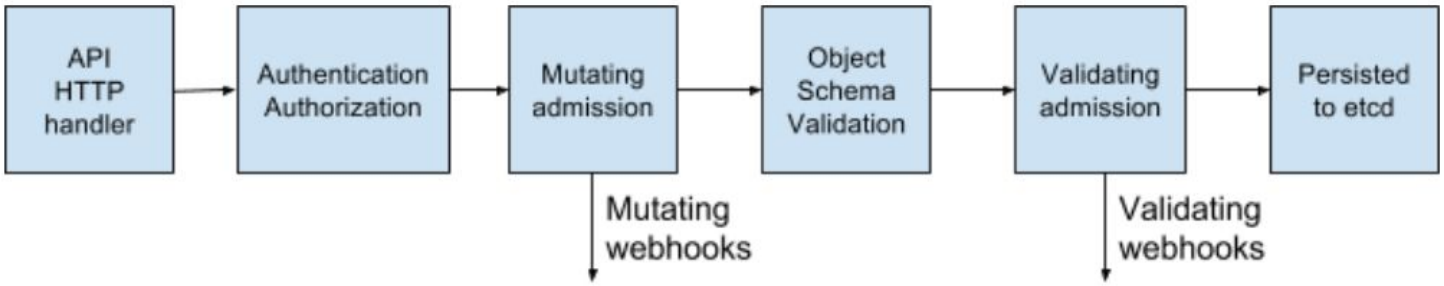
8. Run Containers as a Non-Root User

Containers that run as root frequently have far more permissions than their workload requires which, in case of compromise, **could help an attacker further their attack.**



9. Use Network Policies

By default, Kubernetes networking allows all pod to pod traffic; **this can be restricted using a Network Policy.**



10. Scan Images and Run IDS

Web servers present an attack surface to the network they're attached to: scanning an image's installed files ensures the absence of known vulnerabilities that an attacker could exploit to gain remote access to the container. **An IDS (Intrusion Detection System) detects them if they do.**

11. Run a Service Mesh

A service mesh is a web of encrypted persistent connections, made between high performance "sidecar" proxy servers like Envoy and Linkerd. It adds traffic management, monitoring, and policy - **all without microservice changes.**