# Security Best practice in Kubernetes

Security in Kubernetes is configured at multiple levels. If you are using a cloud provider, it is essential to follow the best practices of cloud provider security, pod-level security standards, and container-level security standards.

**Security Contexts**

Security contexts configure Pods and Containers at runtime. Setting the security context for a Pod ensures that it operates with the necessary security constraints.

A screenshot of a computer

AI-generated content may be incorrect.

**Configure Service Accounts for Pods**

A service account provides an identity for processes that run in a Pod and maps to a ServiceAccount object. When you authenticate to the API server, you identify yourself as a particular user. Kubernetes recognizes the concept of a user, but Kubernetes itself does not have a User API.

**Pull an Image from a Private Registry**

To pull an image from a private registry, create an image secret and use imagePullSecrets. This ensures that private images remain secure and are accessible only to authorized users.

**Use Namespaces**

Namespaces provide a mechanism for isolating groups of resources within a single cluster. Resource names must be unique within a namespace but can be duplicated across different namespaces. Namespace-based scoping is applicable only for namespaced objects such as Deployments and Services, not for cluster-wide objects like StorageClass or PersistentVolumes.

**API Security**

The API endpoints in Kubernetes are secured through Transport Layer Security (TLS), ensuring user authentication with the most secure mechanism available.

**Use Secrets**

A secret is a Kubernetes object that contains sensitive data such as passwords, tokens, or keys, reducing the risk of accidental data exposure. Usernames and passwords are base64-encoded before storage within a Kubernetes cluster. Pods can access secrets at runtime through mounted volumes or environment variables.

**Service Principal**

An Azure service principal is a security identity used by user-created apps, services, and automation tools to access specific Azure resources. It acts as a 'user identity' with specific roles and controlled permissions to access resources securely. Granting only the minimum required permissions improves security.

For example, to access specific Azure resources from a Jenkins pipeline, a service principal (SPN) should be used.

**Service Principal Example:** An administrator can use a service principal with Terraform to build or update an Azure cloud environment without requiring direct credentials.

**Image Pull Secret**

To authenticate and pull images from Azure Container Registry, refer to the following documentation: [Azure Container Registry Authentication](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-auth-kubernetes)

**Dry Run Client**

To create or update a Kubernetes secret if it exists, refer to this guide: [Updating a Kubernetes Secret or ConfigMap](https://blog.atomist.com/updating-a-kubernetes-secret-or-configmap/)

**Authentication**

Kubernetes clusters have two categories of users:

1. **Service Accounts** – Managed directly by Kubernetes.
2. **Normal Users** – Managed by an independent service.

Service accounts created by the Kubernetes API server ensure that every operation managing a process within the cluster is authenticated, reinforcing cluster security.

Applications deployed in Kubernetes can use secrets for secure data access, but secrets are available to all users in the same namespace. To implement proper Role-Based Access Control (RBAC), refer to the official documentation: [RBAC Authentication](https://kubernetes.io/docs/reference/access-authn-authz/rbac/)