# Kubernetes Best Practics

Best practices were categorized into:

1. Building Containers
2. Container Internals
3. Deployments
4. Services
5. Application Architecture

## #1 Building Containers

- Don’t trust arbitrary base images!
- Keep base images small - advantages of smaller images:
  Faster builds
  Less storage
  Image pulls are faster
  Potentially less attack surface
- Use the Builder Pattern

## #2 Container Internals

- Use a non-root user inside the container, Set the Security context runAsNonRoot: true which will make it a policy-wide setting for the entire cluster.
- Make the file system read-only
- One process per container
- Don’t restart on failure. Crash cleanly instead
- Log everything to stdout and stderr

## #3 Deployments

- Use the “record” option for easier rollbacks
- When applying a yaml use the --record flag:
- Use plenty of descriptive labels
- Use sidecars for Proxies, watchers, etc.
- Don’t use sidecars for bootstrapping!
- Don’t use :latest or no tag

## #4 Services

- Don’t use type: LoadBalancer
  Whenever you add load balancer to your deployment file on one of the public cloud providers, it spins one up. This is great for High Availability and speed, but it costs money.

  > Tip: Use Ingress instead which lets you load balance multiple services through a single end-point. This is not only simpler, but also cheaper.

- Use Static IPs they are free!
- Map External Services to Internal Ones

## #5 Application Architecture

- Use Helm Charts
  Helm is basically a repository for packaged up Kubernetes configurations. If you want to deploy a MongoDB. There’s a preconfigured Helm chart for it with all of its dependencies that you can easily use to deploy it to your cluster.
- All Downstream dependencies are unreliable
- Make sure your microservices aren’t too micro
- You want logical components and not every single function turned into a microservice.
- Use Namespaces to split up your cluster
  Role based Access Control