Rolling Updates & Multiple Environments

Rolling Updates with Deployments

- Deployment updates are done through changes to the deployment
- A rolling update is initiated automatically if there's more than 1 replica
- Updates are monitored and stopped if pods fail
- An audit log (history) is kept over all changes
- You can manually roll back to any revision in the history

Update Strategies

- There's 2 update strategies:
    - `Recreate`
    - `RollingUpdate`
- Rolling updates can be adjusted by 2 variables:
    - `maxUnavailable` defines the number (or percentage) of Pods that can be unavailable during the update process.
    - `maxSurge` specifies the maximum number (or percentage) of Pods that can be created above the desired number of Pods.

Blue/Green & Canary Deployments

Blue/Green & Canary Deployments can be implemented in two ways:

- With `kubectl rollout pause` and `kubectl rollout resume`
- With a temporary second Deployment (with fewer replicas)

Multiple Environments & Isolation

How many environments do you need?

- How many stages?
- How many teams?

What level of isolation do you need?

Isolation by Namespaces

Default:

- Isolation is just a default to own namespace
- DNS access over namespaces still possible

Additional Isolation:

- Network Policies
- Admission Rules & RBAC
- Isolation by Nodes (against resource competition)

Isolation by Clusters

- Highest level of isolation
- Management Overhead
- Resource/Cost Overhead
- Spinning up new clusters slower than new namespaces
- Switching clusters a bit harder
- Needed for certain (enterprise) compliance & security requirements

Multi-Environment Best Practices

Keep Configuration out of Images and Deployments -> Config Maps & Secrets

Abstract from actual implementation -> Services