In Kubernetes, rolling updates and rollbacks are crucial for maintaining service availability and ensuring smooth upgrades or downgrades of applications. These features are particularly important for containerized applications, as they allow for zero-downtime deployments and minimize the risk of service disruption.

**1. Rolling Updates:**

* A rolling update is a process in which Kubernetes progressively updates the pods in a deployment, replica set, or stateful set while maintaining the desired number of available instances.
* This is achieved by replacing old pods with new ones, one after another, ensuring that a minimum number of pods are always available to serve traffic.

**Key aspects of rolling updates in Kubernetes:**

1. Selecting a deployment target: Choose the resource (e.g., deployment, replica set, or stateful set) that you want to perform a rolling update on.
2. Defining update strategy: Configure the rolling update strategy using the `RollingUpdate` field in the deployment YAML file. The available strategies are:

\* `RollingUpdate`: The default strategy that progressively updates pods while maintaining the desired number of available instances.

\* `Recreate`: Terminates all existing pods before creating new ones, which can be useful for performing significant updates or resolving issues.

1. Specifying the update configuration: Use the `template` field in the deployment YAML file to define the updated container image or configuration.
2. Triggering the rolling update: Apply the updated deployment YAML file to your Kubernetes cluster. Kubernetes will then start updating the pods based on the defined strategy and configuration.

**2. Rollbacks:**

* A rollback is the process of reverting a deployment to a previous stable state, typically due to issues with the latest update.
* Rollbacks help maintain service availability and minimize the impact of application failures or bugs.

**Key aspects of rollbacks in Kubernetes:**

1. Identifying the problem: Monitor your application and determine that a rollback is necessary. This could be due to increased error rates, performance issues, or other signs of instability.
2. Retrieving the previous revision: In Kubernetes, each deployment maintains a history of previous revisions. You can use the `kubectl rollout undo` command to retrieve the previous revision.
3. Updating the deployment YAML file: Update the deployment YAML file to use the desired previous revision