Module 2:- Workloads and Scheduling

Understand deployments and how to perform rolling update and rollbacks, Use ConfigMaps and Secrets to configure applications, know how to scale applications, Understand the primitives used to create robust, self-healing, application deployments, understand how resource limits can affect Pod scheduling, Awareness of manifest management and common templating tools

Deployments

Deployments are a type of Kubernetes object that lets you manage your applications inside a cluster. They help you run lots of copies of your code (called Pods) and make sure everything is running smoothly. When you make changes to your app, Deployments do the work for you in a safe and smooth way.

Rolling Updates and Rollbacks

Rolling Updates

A rolling update means that instead of stopping your whole app and updating all at once, you update it slowly. The Deployment gradually replaces the old version of your app with the new version, one Pod at a time. This means your app never stops working and users can keep using it without interruptions.

Here's how it works:

1. Kubernetes creates a new ReplicaSet (which is a group of Pods) for the new version.
2. It starts adding the new Pods while slowly removing the old ones.
3. If everything looks good, it keeps replacing the old Pods until only the new version is running.

Rollbacks

Sometimes, the new version of your app doesn't work properly. Maybe there's a bug, or it's too slow. In these cases, you can ask Kubernetes to roll back — switch back to the old version of your app.

There are two ways to rollback:

- **Manual Rollback:** You tell Kubernetes that you want to switch back to the old version.
- **Automatic Rollback:** You can set up rules so that Kubernetes will automatically roll back if something goes wrong.

If you need to rollback, Kubernetes makes sure the old version runs again and removes the new one.

ConfigMaps and Secrets

ConfigMaps

ConfigMaps are used to store configuration data for your apps. They help you keep your app settings separate from your code. This is useful because you can change the settings without changing the code.

For example, you might have a setting for the environment (like “production” or “development”) or a feature flag. You can put these in a ConfigMap and then tell your app to look there for the value.

Secrets

Secrets are like ConfigMaps, but they are for sensitive data like passwords, API keys, and tokens. You don’t want to put these directly in your code or configuration files, so you use Secrets.

Kubernetes stores this information in a secure way, so only your app can access it. You can use Secrets as environment variables or files inside your app.

How to Scale Applications

Manual Scaling

Scaling means changing the number of Pods that are running for your app. You can do this manually by telling Kubernetes how many Pods you want.

For example, you can run this command:

```
kubectl scale deployment/myapp --replicas=10
```

This will make sure 10 Pods are running for your app.

Automatic Scaling

Kubernetes can also scale your app automatically. There are two main ways to do this:

1. **Horizontal Pod Autoscaler (HPA):** This tool watches how much CPU and memory your Pods are using. If usage goes up, it adds more Pods. If usage goes down, it removes some Pods.
2. **Vertical Pod Autoscaler (VPA):** Instead of changing the number of Pods, this changes the resources (CPU and memory) for each Pod. If your app needs more resources, VPA will assign more to each Pod.

How Resource Limits Affect Pod Scheduling

Resource limits are important for making sure your Pods don't use too much CPU or memory. Kubernetes checks these limits when it decides where to put your Pods.

If a Pod doesn't have enough resources available on a node, Kubernetes won't start it. This means your app can't run until there's enough space.

Setting the right limits helps avoid this problem and keeps your cluster running smoothly.

Key Primitives for Self-Healing Deployments

Kubernetes provides native primitives to enable robust, self-healing applications:

- Liveness and Readiness Probes: Detect container health and readiness, automatically restarting unhealthy containers and routing traffic only to healthy ones.
- Pod restartPolicy: Set to Always to ensure containers are restarted automatically on failures.
- ReplicaSets and Deployments: Maintain the desired number of replica Pods, replacing failed ones automatically.

- Automated Recovery and Scaling: Tools like the Horizontal Pod Autoscaler (HPA) and Cluster Autoscaler can automatically adjust the number of Pods and nodes based on resource usage and demand.

Effect of Resource Limits on Pod Scheduling

Resource limits and requests are critical for Pod scheduling:

- Requests: The minimum amount of CPU/memory a Pod needs. The scheduler ensures that nodes have enough allocatable resources to meet all requests for scheduled Pods.
- Limits: The ceiling of resource consumption. If a Pod exceeds its limit, it may be killed by the OOM (out-of-memory) killer for memory, or throttled for CPU.
- Impact on Scheduling: If a node lacks sufficient resources to meet a Pod's request, the scheduler will not assign the Pod, keeping it in the Pending state until resources are available.
- QoS Classes: Pods are classified as BestEffort, Burstable, or Guaranteed based on their resource configurations, which affects eviction priority during resource contention.

Manifest Management and Templating Tools

Kubernetes manifests are declarative YAML/JSON files that define the desired state of applications. Managing these manifests is crucial for automation and consistency:

- Manual Manifests: Use kubectl apply to deploy and manage manifests, storing them in version control for traceability and rollback.
- Templating Tools:
    - Helm: Package and manage complex deployments using charts, with a rich ecosystem of reusable templates.
    - Kustomize: Allows overlaying and patching base manifests, enabling environment-specific configuration customization without duplicating manifests.

- Argo CD and Flux: Enable GitOps workflows, syncing manifests from Git repositories to clusters for automated, auditable deployments.
- Other Tools: Pulumi, Terraform, Carvel, Tanka, and cdk8s offer advanced programming and templating approaches for Kubernetes deployments.

Summary Table: Templating Tools Comparison

Tool	Purpose	Ease of Use	Templating	Deployment	UI/Support
Helm	Package management	★★★	✓	✓	✗
Kustomize	Customization	★★★★	✓	✓	✗
Argo CD	GitOps, Continuous Delivery	★★★★	✗	✓	✓
Flux	GitOps, Continuous Delivery	★★★	✗	✓	✓
Pulumi	Infrastructure as Code	★★★★	✓	✗	✗

Choosing the right tool depends on the complexity, team size, and deployment strategy. dev +1

In summary, Kubernetes' self-healing capabilities rely on robust primitives, smart resource management, and advanced manifest templating tools to ensure resilient, scalable, and easy-to-manage application deployments. plural +2

Summary

- **Deployments** help you manage your app and make updates smoothly.
- **Rolling updates** let you update your app without stopping it.
- **Rollbacks** let you switch back to a previous version if something goes wrong.
- **ConfigMaps** store non-sensitive app settings.
- **Secrets** store sensitive data like passwords and keys.
- You can **scale** your app manually or automatically.
- **Resource limits** help Kubernetes schedule your Pods correctly.
- **Templating tools** help you manage and reuse your manifests.

These concepts are the foundation of managing workloads and scheduling in Kubernetes. They make sure your apps are always running smoothly and securely.

Exercise 2

1. What is the purpose of a rolling update in Kubernetes, and how does it ensure zero downtime during application updates?
2. How can ConfigMaps and Secrets be used to configure applications in Kubernetes, and what is the difference between them?
3. What are the key parameters that control the pace and safety of a rolling update, and how do they affect the update process?
4. Explain how resource limits and requests impact Pod scheduling in Kubernetes, and what happens if a node cannot meet a Pod's resource request?
5. Name two common templating tools used for Kubernetes manifest management and briefly describe their main features.