Storage and Persistence

Volumes and Claims. Configuration Maps and Secrets. Stateful Sets

kubernetes

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

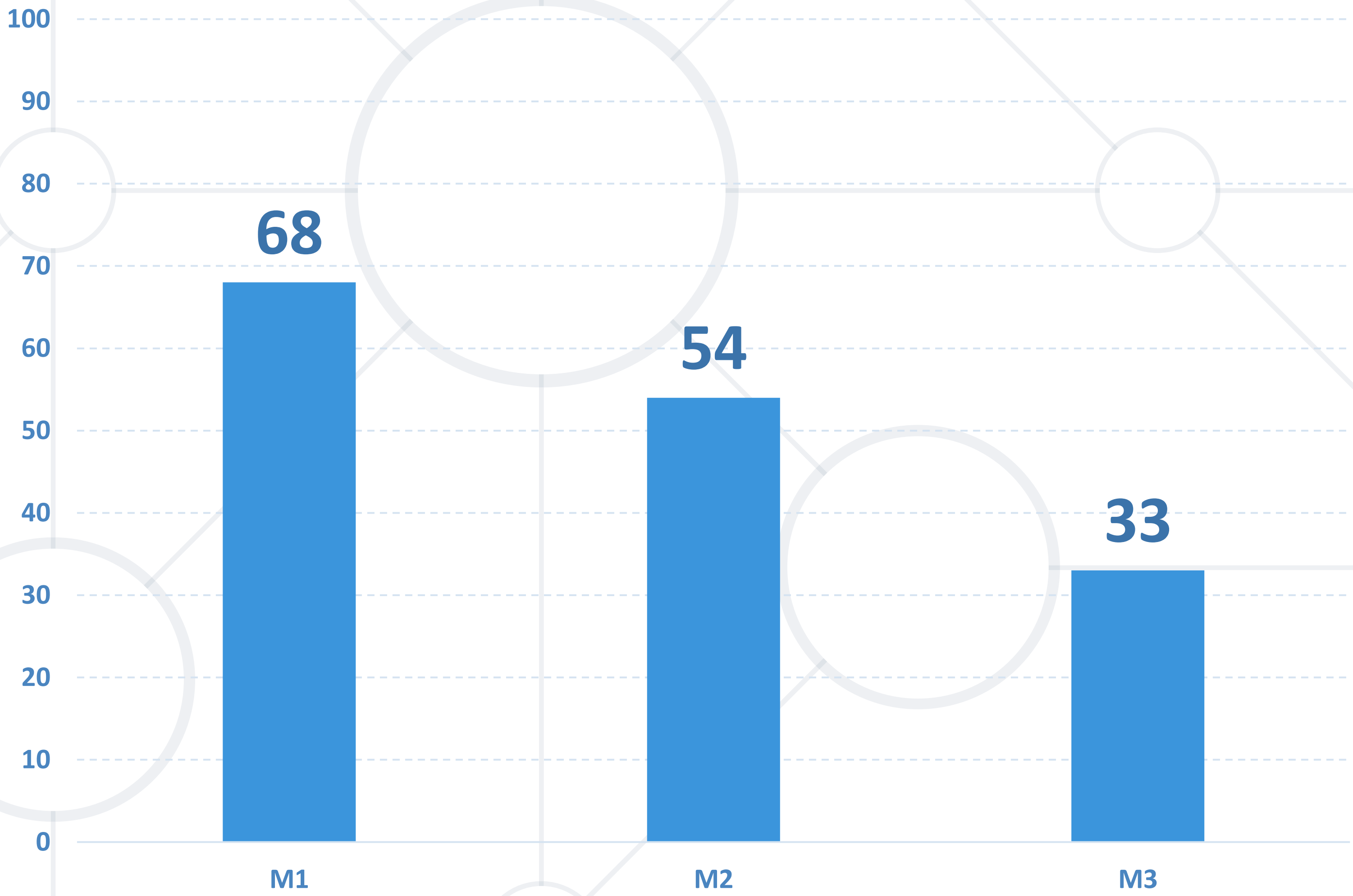
Software University

sli.do

#Kubernetes

facebook.com
/groups/kubernetesnovember2025

Homework Progress (as of 14:00)



Submit M3 until 23:59:59 on 25.11.2025

Submit M4 until 23:59:59 on 02.12.2025

Previous Module (M3)

Quick overview

Table of Contents

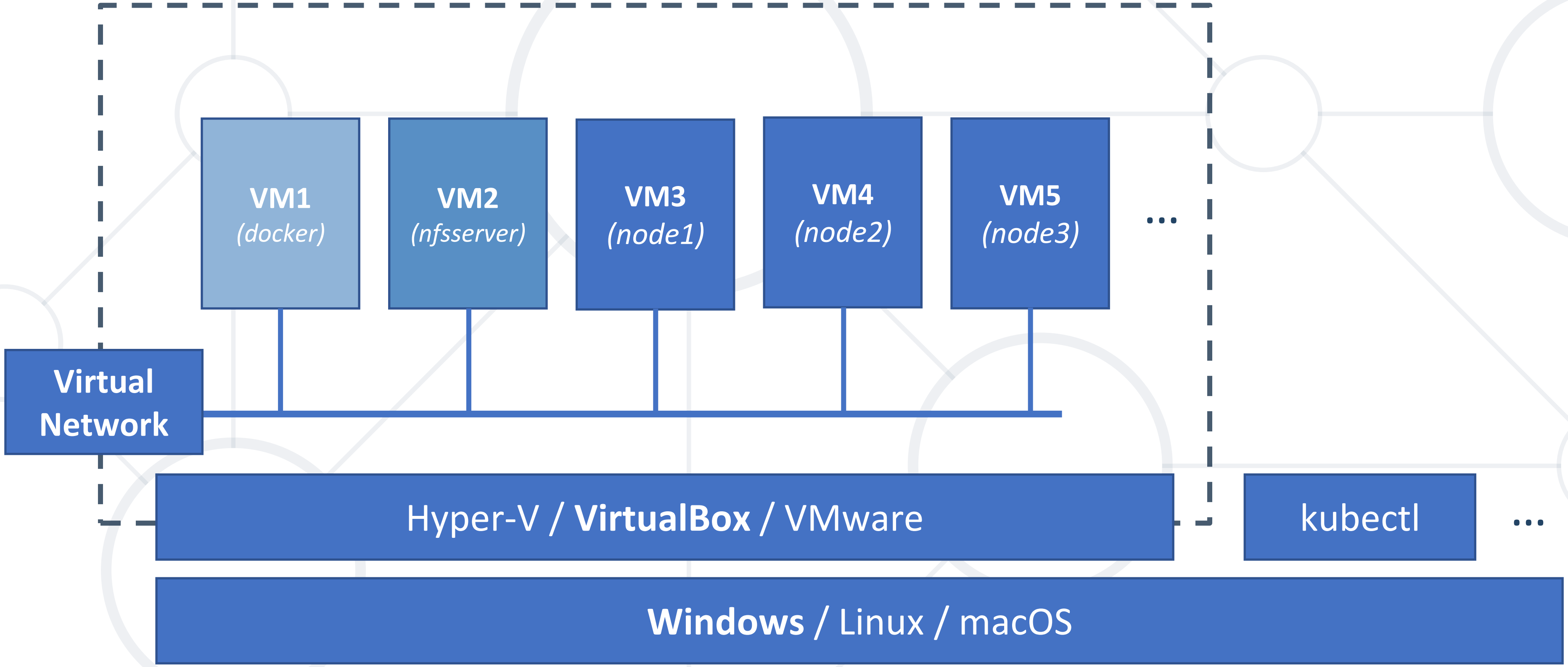
1. Authentication, Authorization and Admission Control
2. Resource Requirements, Limits and Quotas
3. Network Policies

This Module (M4)

Table of Contents

1. (Persistent) Volumes and Claims
2. Configuration Maps and Secrets
3. Stateful Sets

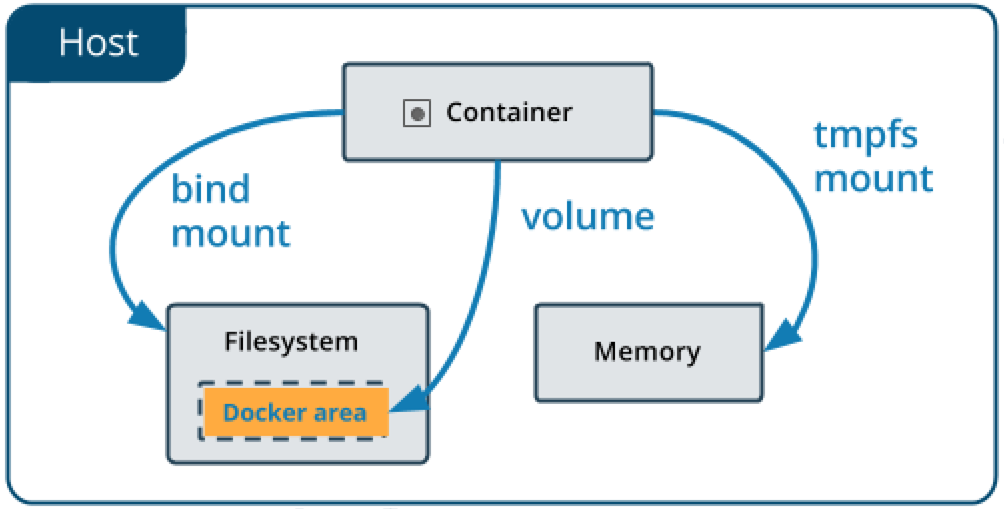
Lab Infrastructure



(Persistent) Volumes and Claims

The Docker Way

- **Bind Mounts** are dependent on the OS and file system structure
- **Volumes** are managed by Docker
- **tmpfs mount** is for non-persistent state data
- **--volume (-v)** is simpler, and **--mount** is more explicit and verbose



https://docs.docker.com/storage/volumes/

Kubernetes Storage Options *

- Volumes
  - Ephemeral
  - Persistent
- Persistent Volumes and Claims
- Storage Classes

https://kubernetes.io/docs/concepts/storage/

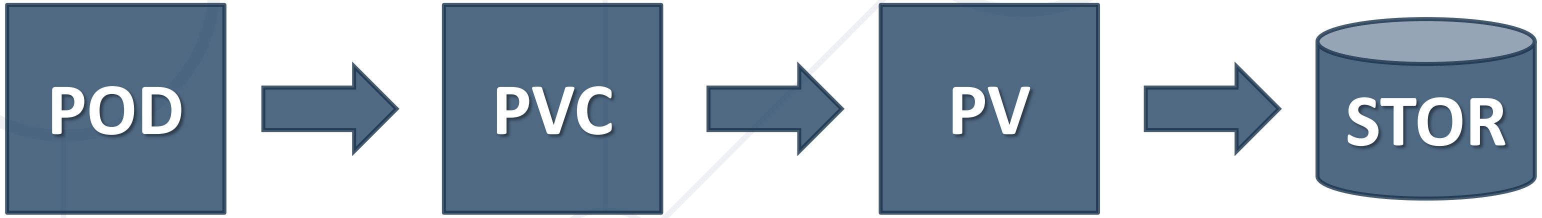
Volumes

- Volumes solve the problem with the **loss of data** when a container is restarted
- In addition, they provide a way to **share data** between containers in a pod
- Support various on-premise options like **nfs**, **cephfs**, **fc**, **iscsi**, etc.
- Support cloud options like **AWS EBS**, **Azure Data Disk**, **GCE PD**, etc.
- They are **declared** (**volumes:**) and **mounted** (**volumeMounts:**) in pod's manifests

https://kubernetes.io/docs/concepts/storage/volumes/

Persistent Volumes and Claims

- Storage administration is a separate activity by itself
- Persistent volumes provide an API that **abstracts** the storage
- **PersistentVolume** (**PV**) is a piece of storage in the cluster that has been provisioned either by the administrator or dynamically
- PVs have an independent lifecycle of the Pods that may use them
- **PersistentVolumeClaim** (**PVC**) is a request for storage by a user



https://kubernetes.io/docs/concepts/storage/persistent-volumes/

PV and PVC Lifecycle (1)

- **Provisioning** can be done either static or dynamic
- **Binding** is the process of **matching** and **attaching** a **PVC** to **PV**. This is done on a **set of criteria**. It is **one-to-one mapping**
- Pods are **using** PVCs as volumes

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

PV and PVC Lifecycle (2)

- When done, PVCs can be deleted. This will trigger the reclaim policy which may be
  - **Retain** allows for manual reclamation of the resource
  - **Delete** removes both the PV object from Kubernetes, as well as the associated storage asset in the external infrastructure
  - **Recycle** performs a basic scrub on the volume and makes it available again (deprecated)

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Storage Classes

- Used to define types or **profiles** of available storage
- Can be used for **automated volume provision**
- Have three main components
  - **Provisioner** determines the volume plugin for PVs provisioning
  - **Parameters** control provisioner’s behavior
  - **Reclaim** policy is inherited by the PVs that will be created

https://kubernetes.io/docs/concepts/storage/storage-classes/

Practice

Live Exercise in Class (Lab)

Configuration Maps and Secrets

Overview

- We may need to inject data into applications
- Effectively, we must pass the data to the containers
- Container runtime offers this via environment variables
- Kubernetes offers
  - Environment variables
  - Configuration Maps
  - Secrets

https://kubernetes.io/docs/tasks/inject-data-application/

Environment Variables

- Key-value pairs used to pass data to the containers inside a pod
- Created in the manifest via **env** or **envFrom** blocks
- **Override** any environment variables in the container
- They may **reference** each other but then their **definition order is important**
- Reference is done via the **$(REFVAR)** construct

https://kubernetes.io/docs/tasks/inject-data-application/define-environment-variable-container/

Configuration Maps

- Used to store **non-confidential data** in key-value pairs
- Pods can consume them as **environment variables**, **command-line arguments**, or as **configuration files in volumes**
- We use them to **separate configuration data** from **application code**
- They are not designed to store large chucks of data (**max 1 MiB**)
- The name of a **ConfigMap** must be a **valid DNS subdomain** name

https://kubernetes.io/docs/concepts/configuration/configmap/

Secrets

- Contain a small amount of sensitive data such as a **password**, a **token**, or a **key**
- This way confidential data is separated from the application code
- Similar to **ConfigMaps** but are specifically intended to hold confidential data
- Consumed via **files in a volume**, **environment variables**, or by the kubelet while pulling images for the pod (**imagePullSecrets**)
- Secrets can be **opaque**, **tls**, **token**, **service-account-token**, etc.

https://kubernetes.io/docs/concepts/configuration/secret/

Practice

Live Exercise in Class (Lab)

Stateful Sets

Stateful Sets

- Used to manage **stateful applications**
- Manage the deployment of a **set of Pods**
- Pods are with identical container specifications just **like** with **Deployment** and **ReplicaSet**
- The main **difference** here is that the Pods have **persistent identifiers** that are **maintained across rescheduling**
- **Storage volumes** can be used as part of the solution for providing persistence

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Stateful Sets Added Value

- Stable and unique network identifiers
- Stable and persistent storage
- Ordered graceful deployment and scaling
- Ordered and automated rolling updates

***** Stable = persistence across pod (re)scheduling**

***** Ordered = when scaling up it is done from 0 to N and when scaling down it is done from N to 0**

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Stateful Sets Limitations

- Storage for a Pod must be provisioned **upfront** either automatically or by an administrator
- Deleting or scaling down **doesn't delete the associated volumes**
- **Headless service** is required for the network identity of the pods
- StatefulSet deletion **doesn't guarantee the pods termination order.** If required, first we must scale it down to 0
- Rolling updates (with OrderedReady policy) may get broken and then a manual intervention may be required

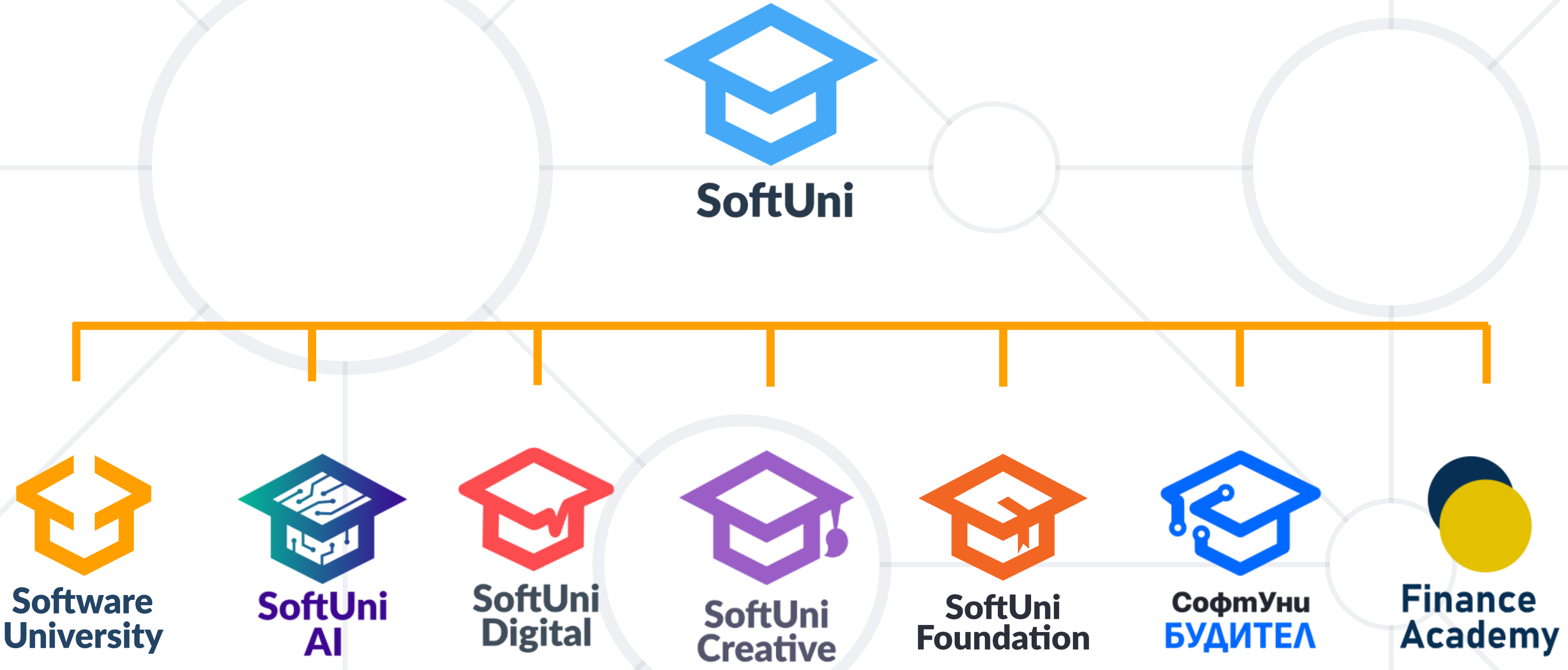
https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Practice

Live Exercise in Class (Lab)

Software University

Questions?



SoftUni

Software University

SoftUni AI

SoftUni Digital

SoftUni Creative

SoftUni Foundation

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

VIVACOM

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License