# Security and Policies

## Authentication, Authorization, and Admission.

## Resource Management. Network Policies

kubernetes


**SoftUni Team**

**Technical Trainers**

Software University


SoftUni


**Software University**

https://softuni.bg

# Have a Question?

Software University

## sli.do

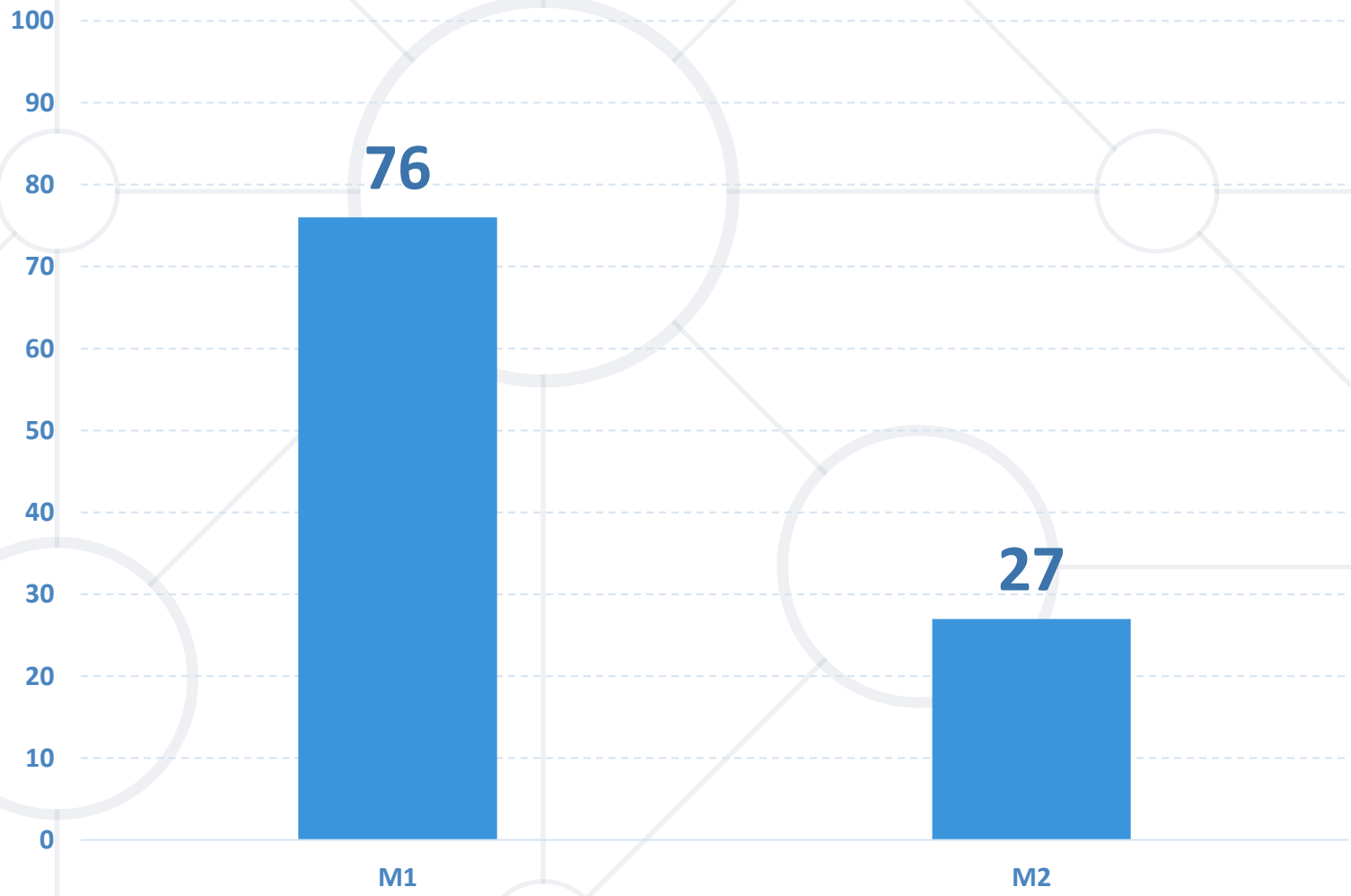
## #Kubernetes

---

## https://www.facebook.com/groups

## /KubernetesOctober2023

# Homework Progress




Submit M2
until 23:59:59
on 30.10.2023

Submit M3
until 23:59:59
on 06.11.2023

# Previous Module (M2)

## Quick overview

# Table of Contents



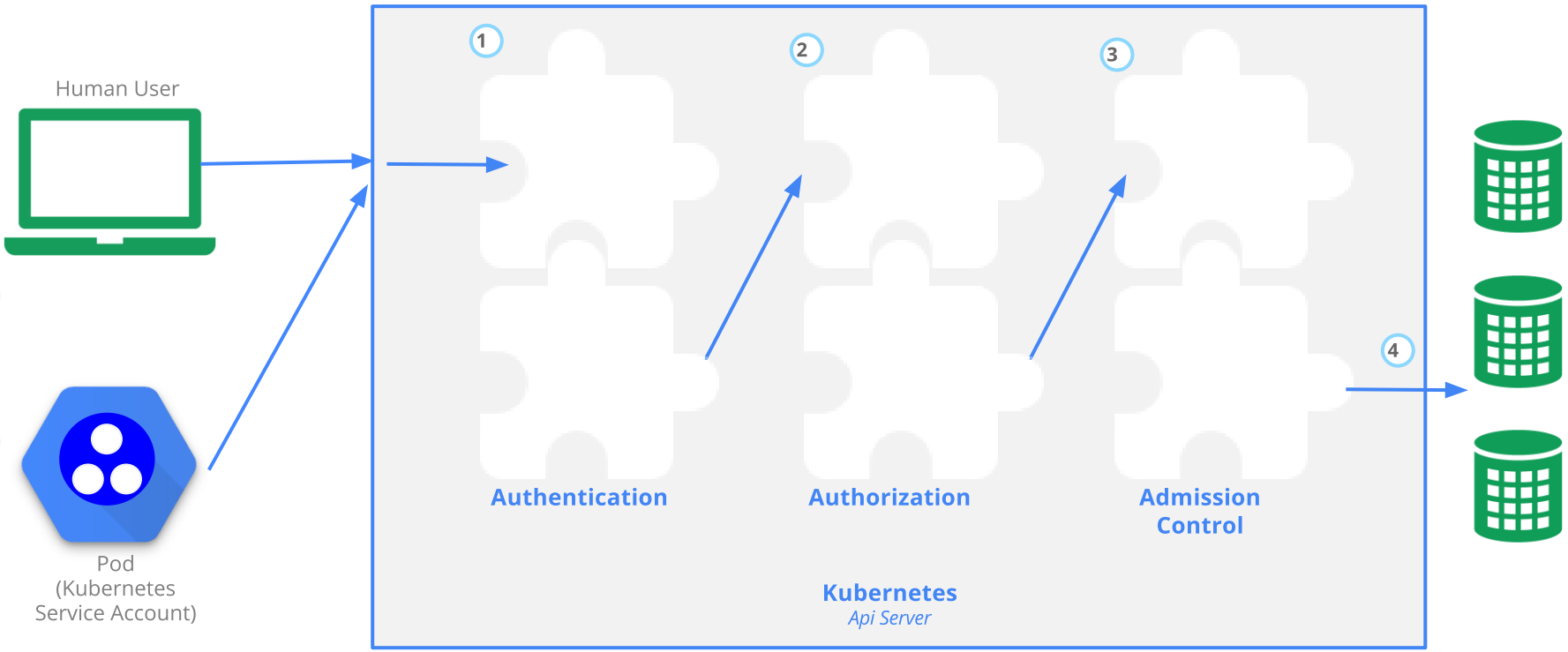
# This Module (M3)

# Table of Contents



# Authentication, Authorization and Admission Control

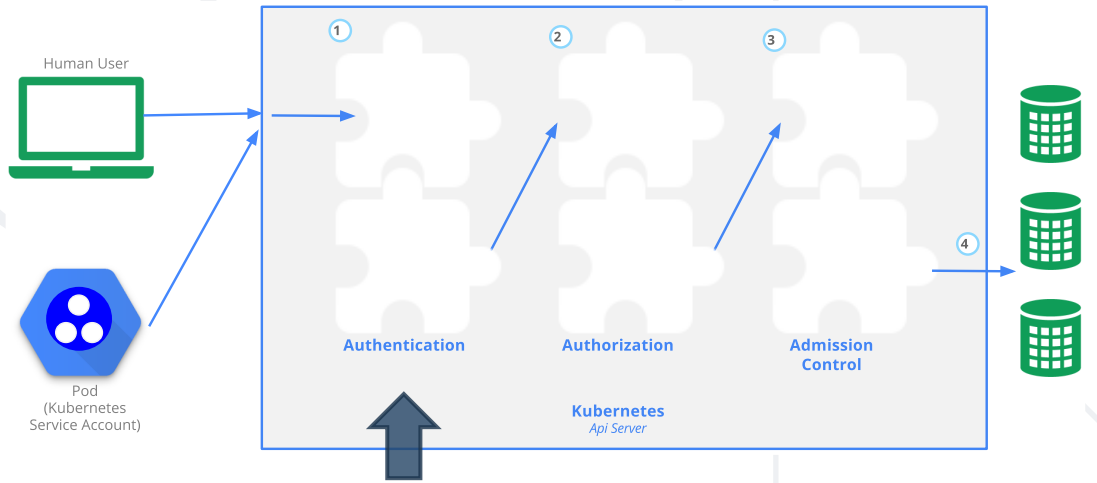
# Access Control Overview




https://kubernetes.io/docs/concepts/security/controlling-access/

# Authentication

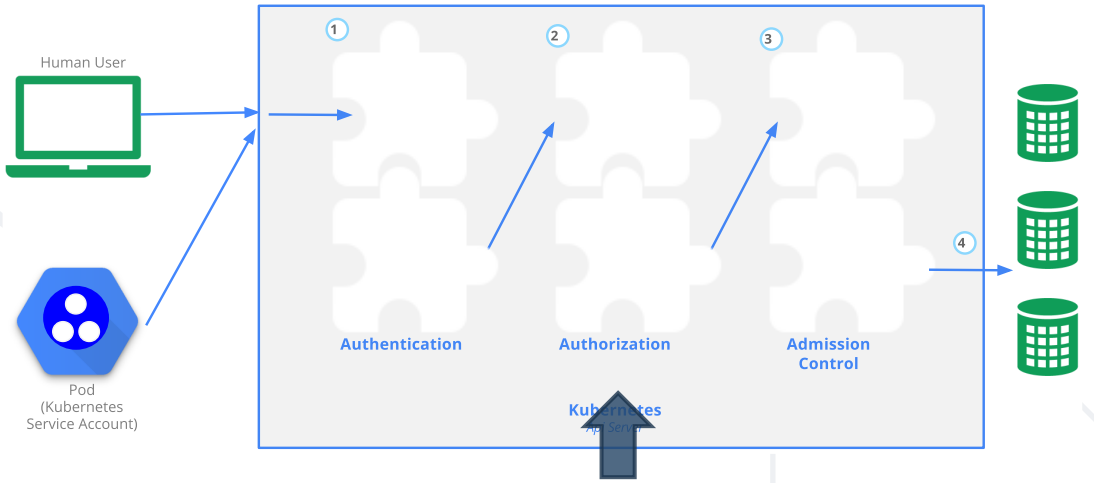
- Authentication modules include **client certificates**, **password**, and **plain tokens**, **bootstrap tokens**, and **JSON Web Tokens**
- Multiple authentication modules can be specified. Each one is tried in sequence, until one of them succeeds
- If the request cannot be authenticated, it is rejected with HTTP status code 401
- If process succeeds, the user is authenticated as a specific **username**, and it is available to subsequent steps to use in their decisions




https://kubernetes.io/docs/concepts/security/controlling-access/

# Authorization

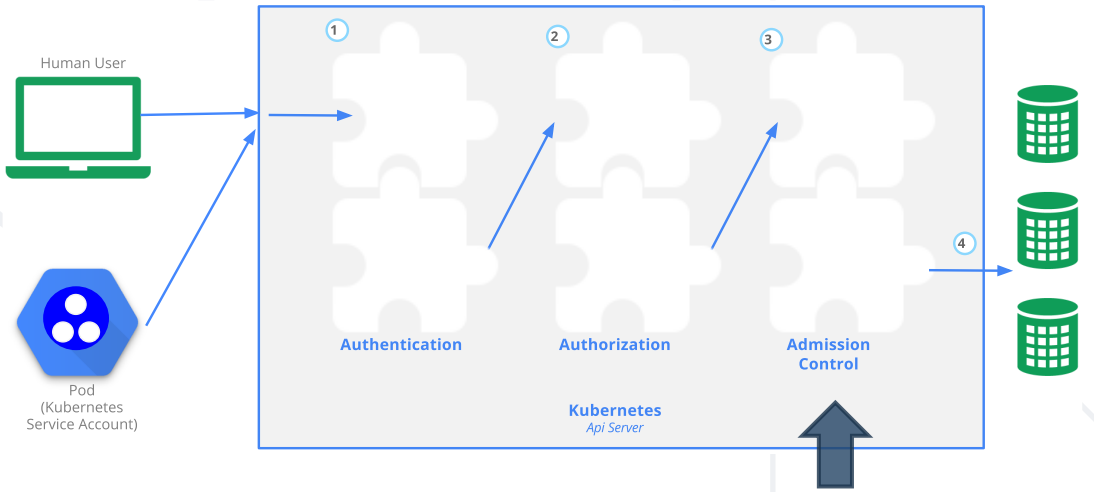
- After a successful authentication, the request must be authorized
- It must include the **username**, the **action**, and the **target object**
- It is authorized if an existing policy confirms that the user has the right to complete the requested action
- Multiple authorization modules are supported, such as **ABAC**, **RBAC**, and **Webhook**
- They are configured during the cluster creation
- More than one may be specified
- All are evaluated until one authorizes the request
- If all deny it, then the request is denied (403)




https://kubernetes.io/docs/concepts/security/controlling-access/

# Admission Control

- Software modules that can **modify** or **reject** requests
- In addition to the attributes available to Authorization modules, Admission Control modules can access the **contents of the object** that is being created or modified
- **Multiple controllers** may be configured. In this case they are **called in order**
- If any admission controller module rejects, then the request is immediately rejected
- They act on requests that **create**, **modify**, **delete**, or **connect to** (proxy) an object
- They **ignore** requests that merely **read objects**
- They can also set complex defaults for fields




https://kubernetes.io/docs/concepts/security/controlling-access/

# User Accounts vs Service Accounts

Software University

- When we (humans) access the cluster (for example with kubectl), we are authenticated by the **apiserver** as a particular **User Account** (usually **admin**)
- Processes in containers inside pods can also contact the **apiserver**. When they do, they are authenticated as a particular **Service Account** (for example, **default**)

https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

# Service Accounts

- Each namespace gets one by default named **default**
- During pod creation we can specify a service account
- If we omit it, the default one is assigned
- Service account names are formatted like this:
  **system:serviceaccount:<namespace>:<service account name>**
- We can create additional ones to further adjust pods' rights
- Pods can use service accounts from the same namespace
- Each pod is attached to only one service account
- One service account may be shared by multiple pods

https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

# Role-based Access Control (RBAC)

Software University

- Controls who can do what with which type of resources
- **Roles** contain rules that represent a set of permissions
- Permissions are purely additive (there are no "deny" rules)
- A **role binding** grants the permissions defined in a role to a user or set of users
- Role bindings hold a list of **subjects** (users, groups, or service accounts), and a **reference to the role** being granted

https://kubernetes.io/docs/reference/access-authn-authz/rbac/

# Roles and Bindings

- **Roles** set permissions within a namespace
- **ClusterRoles** are non-namespaced resources. They are used to
  - Grant permissions on namespaced resources within individual namespaces
  - Grant permissions on namespaced resources across all namespaces
  - Grant permissions on cluster-scoped resources

https://kubernetes.io/docs/reference/access-authn-authz/rbac/

# Roles and Bindings

- **RoleBinding** grants permissions within a specific namespace whereas a **ClusterRoleBinding** grants that access cluster-wide
- RoleBinding may reference any Role in the same namespace or a ClusterRole
- If we want to bind a ClusterRole to all namespaces, we must use ClusterRoleBinding

https://kubernetes.io/docs/reference/access-authn-authz/rbac/

# Practice

Live Exercise in Class (Lab)

# Resource Requirements, Limits and Quotas

# Resource Requests and Limits

Software University

- By default, containers run with unbounded resources on a cluster
- We can control how much of a resource is granted to a Container
- Most common resources are the CPU and memory
- We specify how much resources a Container needs to operate via the **request** option. This information is accumulated on a Pod level and a decision for scheduling of the Pod is made
- We restrict how much of a resource a Container can use via the **limit** option

https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

# CPU

Software University

- The CPU resource is measured in **CPU units**
- One CPU, in Kubernetes, is equivalent to **1 AWS vCPU** or **1 GCP Core** or **1 Azure vCore** or **1 Hyperthread on a bare-metal Intel processor with Hyperthreading**
- We can specify fractional values, for example **0.5** which is the half of 1 CPU
- The above can be stated as **500m** (milicpu) units

https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource/

# Memory

Software University

- Memory requests and limits are measured in **bytes**
- We can specify it using **integer** values or **fixed-point** numbers
- We may use the standard suffixes **k**, **M**, **G**, **T**, etc. or the power-of-two ones **Ki**, **Mi**, **Gi**, **Ti**, etc.
- For example, to set **128 MiB** of memory, we must use **128Mi**

https://kubernetes.io/docs/tasks/configure-pod-container/assign-memory-resource/

# Quotas

Software University

- Resource quotas are a tool for administrators to address the need of sharing cluster resources between teams of users
- They are defined by the **ResourceQuota** object
- Provide constraints that **limit aggregate resource consumption per namespace**
- Limit the **quantity of objects** that can be created in a namespace by type
- Limit the **total amount of compute resources** that may be consumed by resources

https://kubernetes.io/docs/concepts/policy/resource-quotas/

# Limit Ranges

Software University

- A LimitRange is a policy to constrain resource allocations (to Pods or Containers) in a namespace
- Enforce minimum and maximum compute resources usage per Pod or Container in a namespace
- Enforce minimum and maximum storage request per PersistentVolumeClaim in a namespace
- Enforce a ratio between request and limit for a resource in a namespace
- Set default request/limit for compute resources in a namespace and automatically inject them to Containers at runtime

https://kubernetes.io/docs/concepts/policy/limit-range/

# Practice

Live Exercise in Class (Lab)

# Network Policies

# Network Policies

Software University

- Control traffic flow at the IP address or port level for particular applications in the cluster
- Allow control if and how a pod is allowed to communicate with various network entities over the network
- The entities are identified by a combination of
  - Other pods that are allowed
  - Namespaces that are allowed
  - IP blocks

https://kubernetes.io/docs/concepts/services-networking/network-policies/

# Prerequisites

- They are **implemented by the network plugin**
- **Flannel** does not support Network Policies
- Plugins like **Calico**, **Weave Net**, and **Cilium** do support them
- Creating a NetworkPolicy resource without a plugin that supports it won't have any effect

https://kubernetes.io/docs/concepts/services-networking/network-policies/

# Main Elements

Software University

- **podSelector** selects a pod or group of pods. If an empty one is used this matches all pods in the namespace
- **policyTypes** can include either **Ingress** or **Egress** or both. It specifies which traffic for the selected pods is being regulated
- **ingress** includes a list of allowed ingress rules. Each rule consists of **from** and **ports** sections
- **egress** includes a list of allowed egress rules. Each rule consists of **to** and **ports** sections

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
```

https://kubernetes.io/docs/concepts/services-networking/network-policies/

# Default Policies

- By default, no policies exist in a namespace, so all ingress and egress traffic is allowed
- Should we want, we can create the following ***default*** network policies

**Deny all ingress traffic**

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

**Allow all ingress traffic**

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-ingress
spec:
  podSelector: {}
  ingress:
  - {}
  policyTypes:
  - Ingress
```

**Deny all egress traffic**

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-egress
spec:
  podSelector: {}
  policyTypes:
  - Egress
```

**Allow all egress traffic**

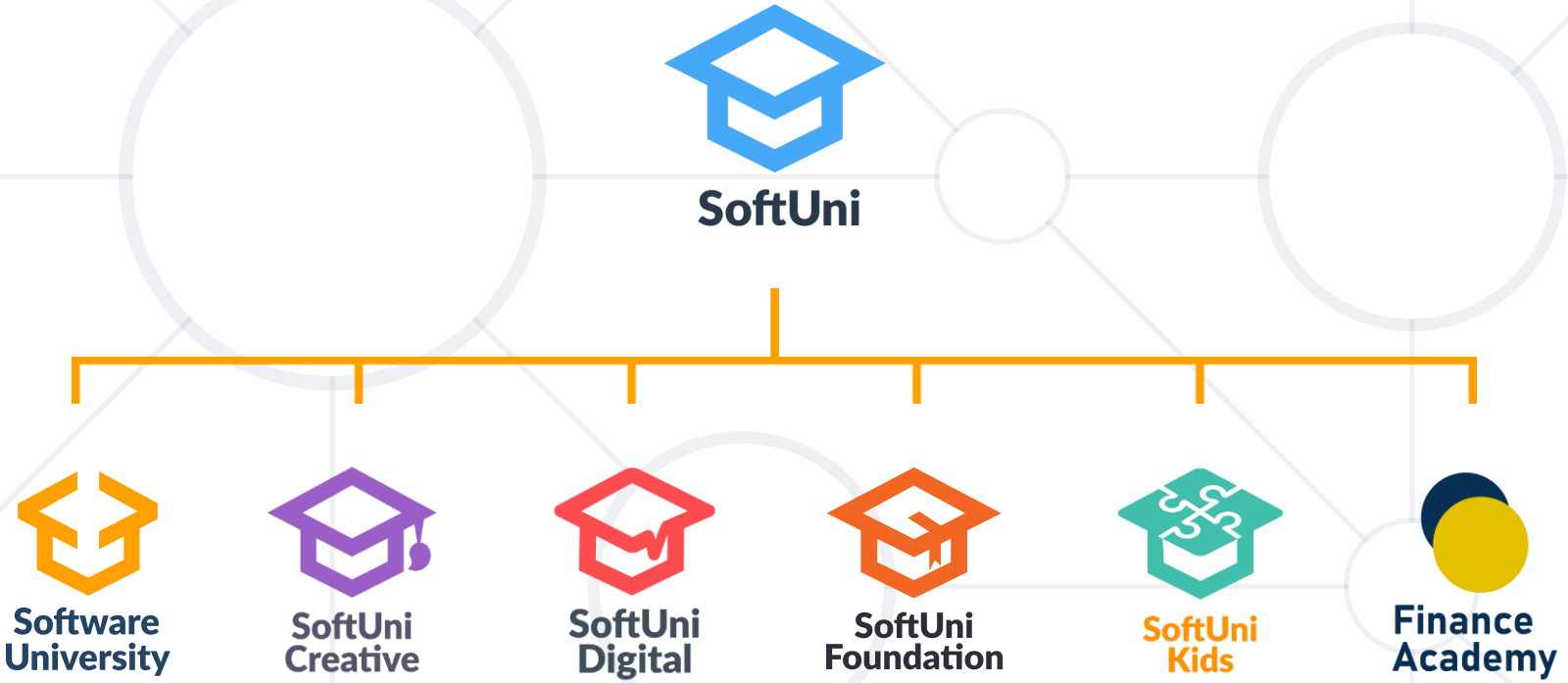
```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress
spec:
  podSelector: {}
  egress:
  - {}
  policyTypes:
  - Egress
```

https://kubernetes.io/docs/concepts/services-networking/network-policies/

# Practice

Live Exercise in Class (Lab)

# Questions?



# SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

# License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

# 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
- Software University Forums
  - forum.softuni.bg