

# Final Summary

# Kyma Runtime

# Kubernetes Features

Kubernetes has several features that allow you to manage containerized applications in a scalable and reliable way. Among the most important features are the following:

## Service discovery and load balancing

Containers can be exposed via their DNS name or using their IP address. Adopting a load balancing approach, Kubernetes can distribute network traffic to stabilize deployment when container traffic is too high.

## Storage orchestration

Storage can be automatically integrated with Kubernetes using a storage system of your choice, such as on-premises storage, public cloud providers, or a networked storage system.

## Automated rollouts and rollbacks

Kubernetes allows you to update containers using rolling updates. Kubernetes starts new containers and then kills old ones. This feature also ensures zero downtime during the deployment. Should anything go wrong during the rollout process, Kubernetes rolls back the change for you.

## Immutability and self-healing

Everything in Kubernetes is defined declaratively. You specify the desired state of an application or infrastructure, and Kubernetes always ensures that the current state matches the desired state. When a container crashes, Kubernetes restarts it for you. And when an entire node goes down, Kubernetes moves the containers to a healthy node and shuts down containers that don't respond to your custom health checks.

## Secret and configuration management

You can use Kubernetes to store and manage confidential information, such as passwords, tokens, and SSH keys. You can deploy and update secrets along with application configurations without rebuilding your container images or revealing sensitive information in applications.

## Extensibility and Ecosystem

The Kubernetes API comes with a standard set of implementations and objects. Kubernetes often defines the API but not the concrete implementation. This allows you to use and switch between different implementations of the same API. For example, you can use different storage systems for persistent volumes. You can also use different container runtimes, such as Docker, rkt, and Rocket. Over time, the Kubernetes ecosystem has grown to a size where you can find a solution for almost any problem. For example, there are solutions for logging, monitoring, service meshes, ingress, and many more. We will cover some of them later in this course.

# Kubernetes API via kubectl

**Kubectl syntax is as follows**

*kubectl [command] [TYPE] [Name] [flags]*

*kubectl get pods*

*kubectl get pods --namespace my-namespace*

*kubectl get services*

*kubectl get replicases*

# Kubernetes API via kubectl

**Q1.** Which command can you use to create a pod from a YAML file?

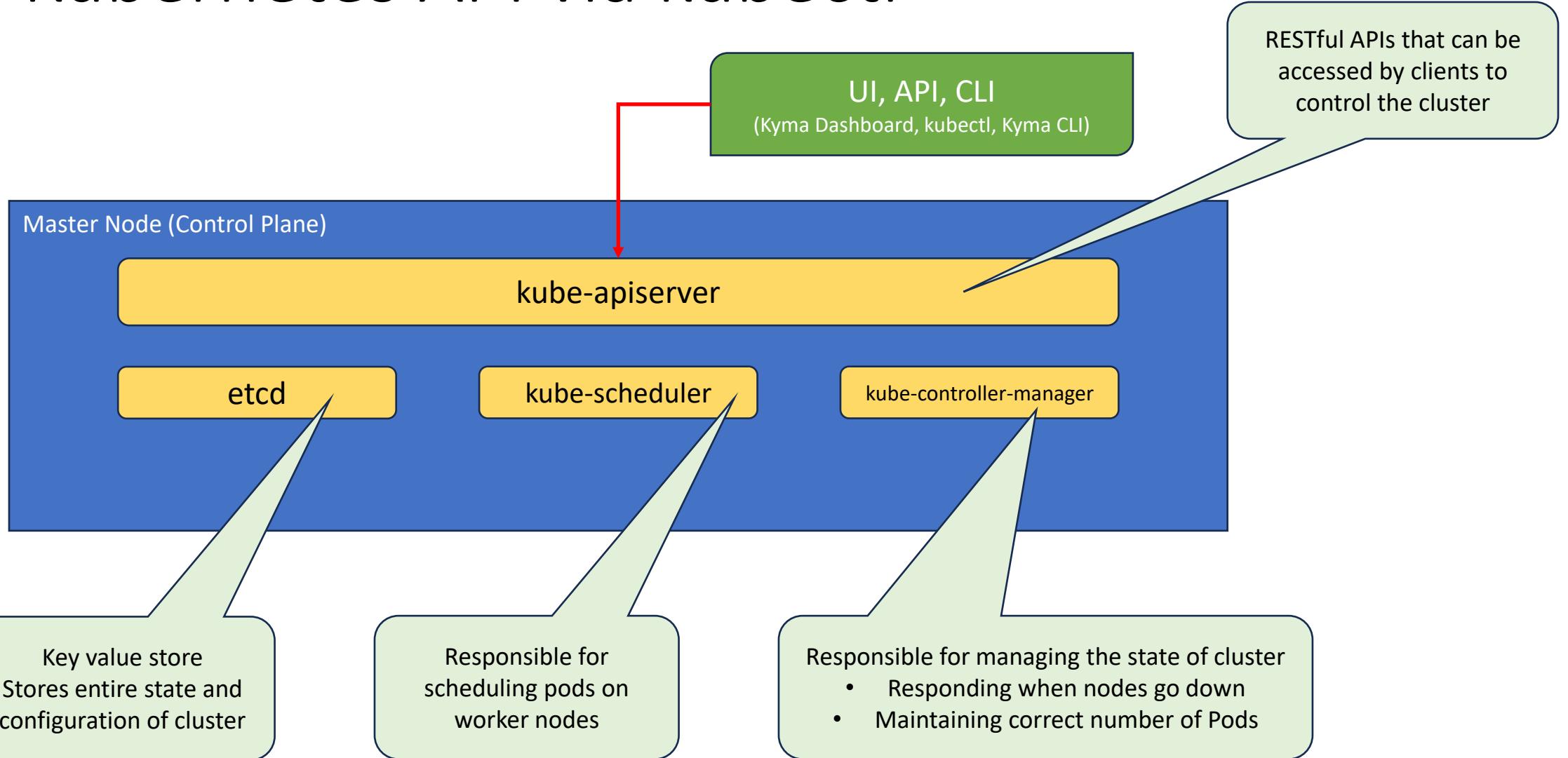
- A `kubectl create -f my-pod.yaml`
- B `kubectl create my-pod.yaml`
- C `kubectl upload my-pod.yaml`



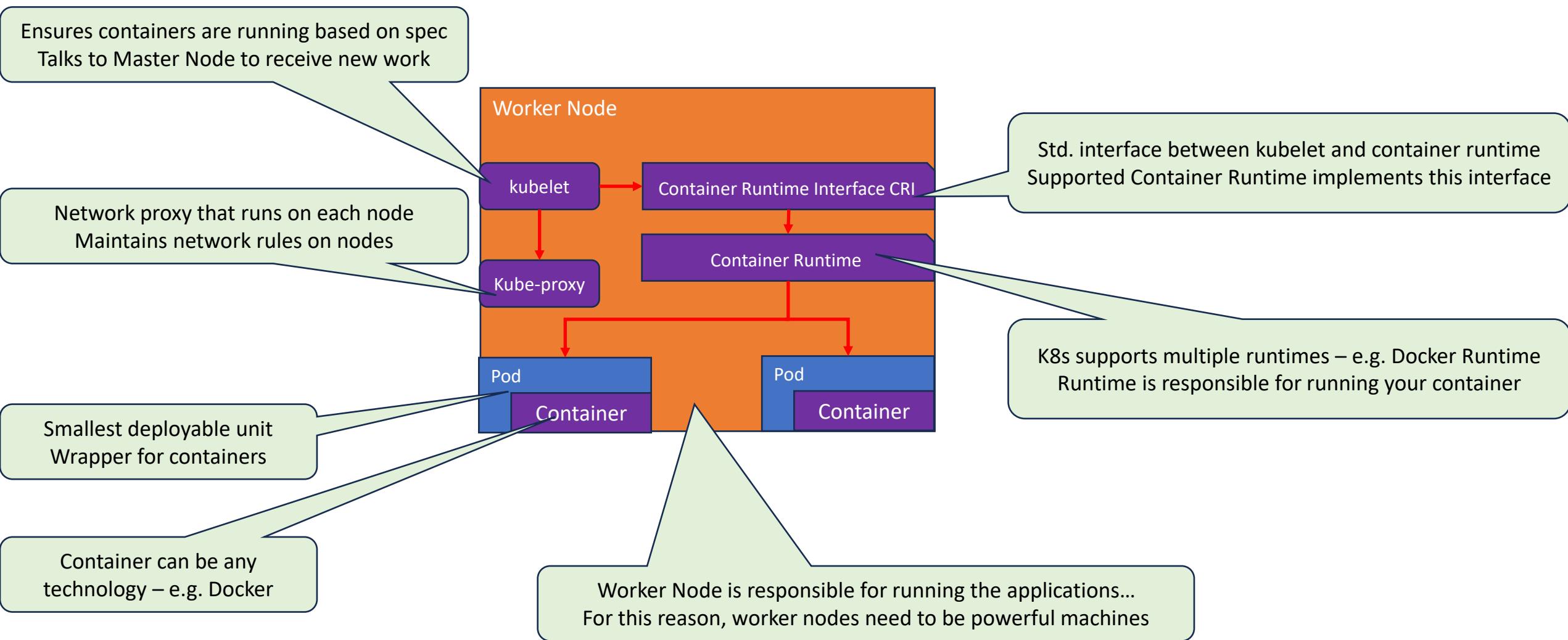
Correct

Correct. To create a pod from a YAML file you use the command:`kubectl create -f my-pod.yaml`.

# Kubernetes API via kubectl



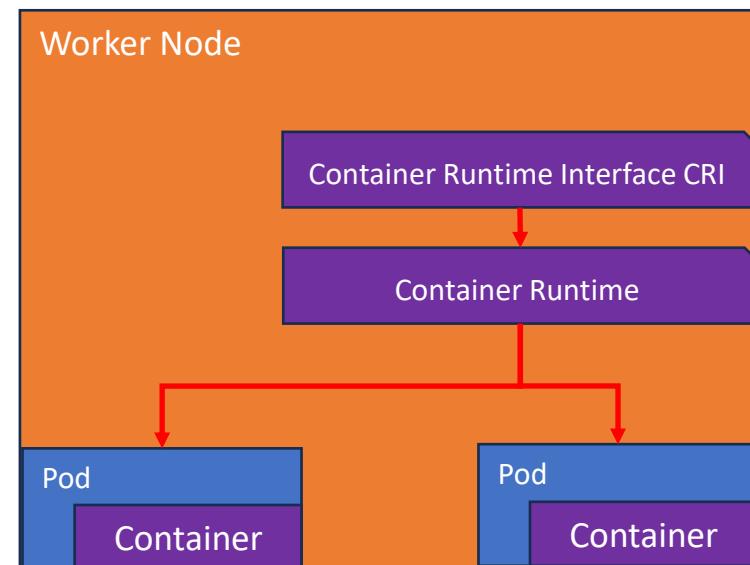
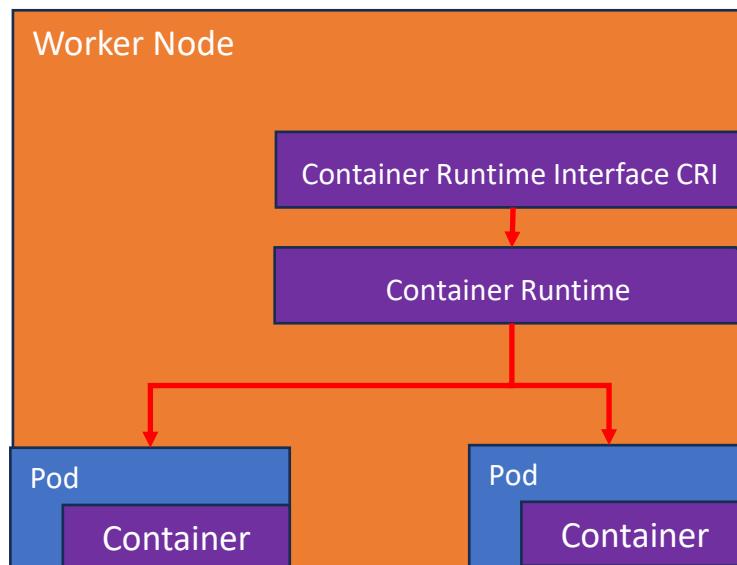
# Kubernetes API via kubectl



# Kubernetes Cluster

K8s does not want to be tied to a single container technology (for example, Docker)

K8s allows you to use any container runtime that implements this Container Runtime Interface



# Kubernetes Components

For high availability, we typically have more than 1 Application Pod

Do we need to define multiple Application Pods in the code ? **No**

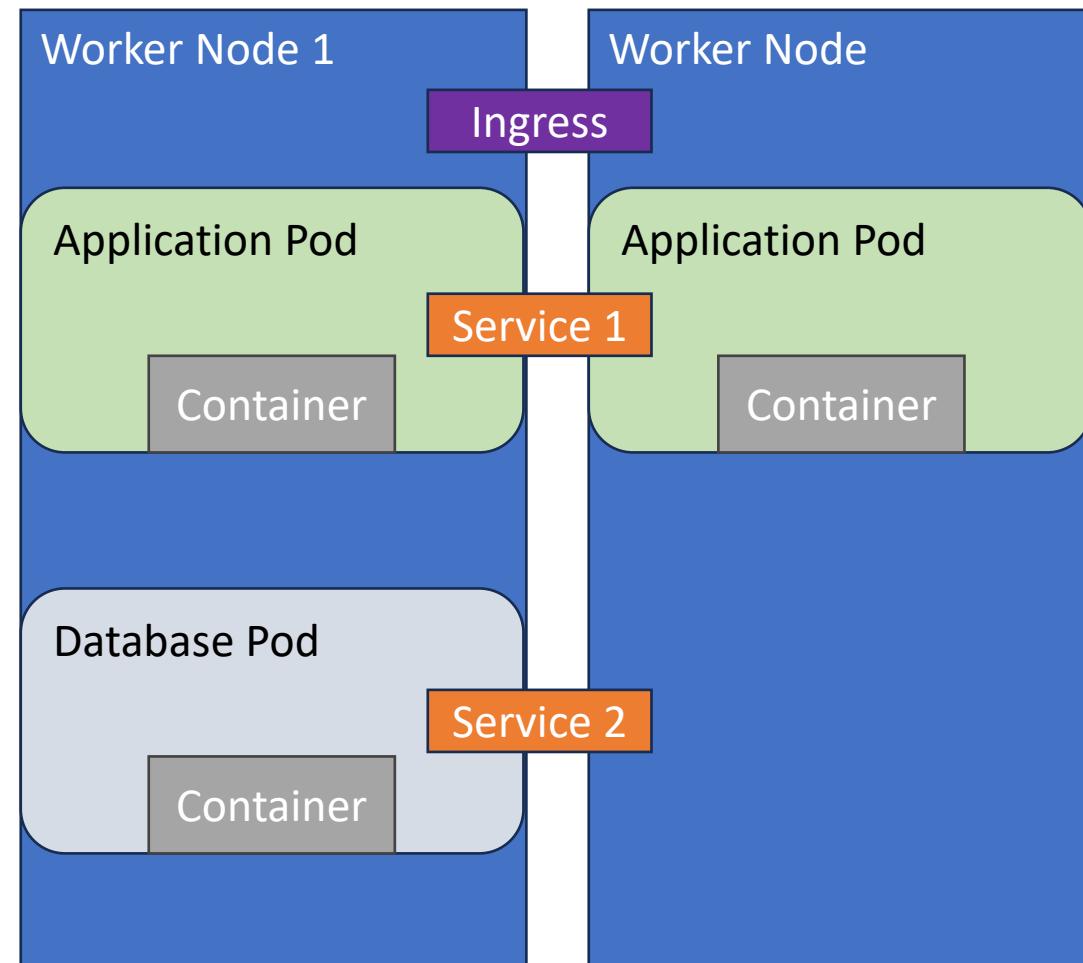
**ReplicaSets** define how many Pods we need in the cluster

Service can also act as a load balancer

Deployments specify how many replicas you need

## Layers of abstraction

- Deployment manages a ReplicaSet
- ReplicaSet manages a Pod
- Pod is an abstraction of a Container



# Exposing a Service

**Q1.** How can you securely expose a service?

A By creating a Kubernetes service with port 443

By creating an API Rule with Access Rules



Correct

Correct. You can securely expose a service by creating an API Rule with Access Rules.

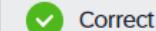
**Q3.** What are services in Kubernetes?

A Services are a way to group StatefulSets together

Services are a way to group Pods together

C Services are a way to group Deployments together

D Services are a way to group ReplicaSets together



Correct

Correct. The services in Kubernetes are a way to group Pods together.

# Serverless Functions

The screenshot shows the Kyma Platform UI with the following details:

- Header:** Kyma (with a logo), a dropdown menu, and a "default" namespace selector.
- Left Sidebar:** A navigation tree with sections:
  - Workloads:** Functions (highlighted with a red circle containing '1'), Deployments, Stateful Sets, Daemon Sets, Cron Jobs, Jobs, Replica Sets, and Pods.
  - Discovery and Network:** API Rules (highlighted with a red circle containing '2'), Ingresses, Services, Horizontal Pod Autoscalers, and Network Policies.
  - Istio:** Service Management, Storage, Apps, and Configuration.
- Central Content:**
  - Default Namespace Overview:** Labels (istio-injection=enabled, kubernetes.io/metadata.name=default), Created (17 days ago), Status (ACTIVE), and buttons for Edit, Deploy new workload +, Upload YAML +, and Delete.
  - Healthy Resources:** Pods (0/0) and Deployments (0/0).
  - Resource Consumption:** Memory Requests (0/0) and Memory Limits (0/0).
  - Metrics:** A chart titled "Virtual service is created by default" showing CPU usage over time (18:07 to 19:06). The chart indicates 1 virtual service with 0 CPU requests. A legend shows a purple square for CPU.
  - Limit Range:** A table with columns: Name, Created, Labels, Max, Min, Default, and Default Request. It displays the message "No entries found".
  - Resource Quota:** A table with columns: Name, Created, Labels, Max, Min, Default, and Default Request. It displays the message "No entries found".

# Deployments

The screenshot shows the Kyma Platform interface with the following details:

- Left Sidebar:** A navigation menu with the following sections:
  - Workloads
    - Deployments (marked with a red circle containing '1')
    - Stateful Sets
    - Daemon Sets
    - Cron Jobs
  - Jobs
    - Replica Sets
  - Pods
  - Discovery and Network
    - API Rules (marked with a red circle containing '3')
    - Ingresses
    - Services (marked with a red circle containing '2')
    - Horizontal Pod Autoscalers
  - Network Policies
  - Istio
    - Service Management
    - Storage
    - Apps
    - Configuration

**Top Bar:** Shows the Kyma logo and a dropdown menu labeled "default".

**Main Content Area:** Displays the following information:

- Labels:** `istio-injection=enabled`, `kubernetes.io/metadata.name=default`
- Created:** 17 days ago
- Status:** ACTIVE
- Edit**, **Deploy new workload +**, **Upload YAML +**, **Delete** buttons.
- Healthy Resources:** Pods 0/0, Deployments 0/0
- Resource Consumption:** Memory Requests 0/0, Memory Limits 0/0
- CPU Usage Graph:** A line chart showing CPU usage over time. The Y-axis ranges from 0 to 1. The X-axis shows times: 18:07, 18:22, 18:37, 18:52, 19:06. A single data point is shown at 18:37 with a value of 1.837. The legend indicates "CPU". Time intervals: 1h, 3h, 6h.
- Limit Range:** A table with columns: Name, Created, Labels, Max, Min, Default, Default Request. It displays the message "No entries found".
- Resource Quota:** A table with columns: Name, Created, Labels, Max, Min, Default, Default Request. It displays the message "No entries found".

# Exposing a Service

**Q5.** How does a service determine which pods to group together?

- A By using a port selector
- B By using a field selector
- C By using a label selector
- D By using a hostname selector

 Correct

Correct. A service determines which pods to group together by using a label selector.

**Q5.** Which of the following statements are an example of how Istio is integrated into project "Kyma"?

- A Istio is installed but not enabled by default.
- B Istio's components are installed in the istio-system namespace.
- C The APIRule custom resource is translated into Istio components (VirtualService).
- D A Kubernetes Ingress translates into Istio components (VirtualService).

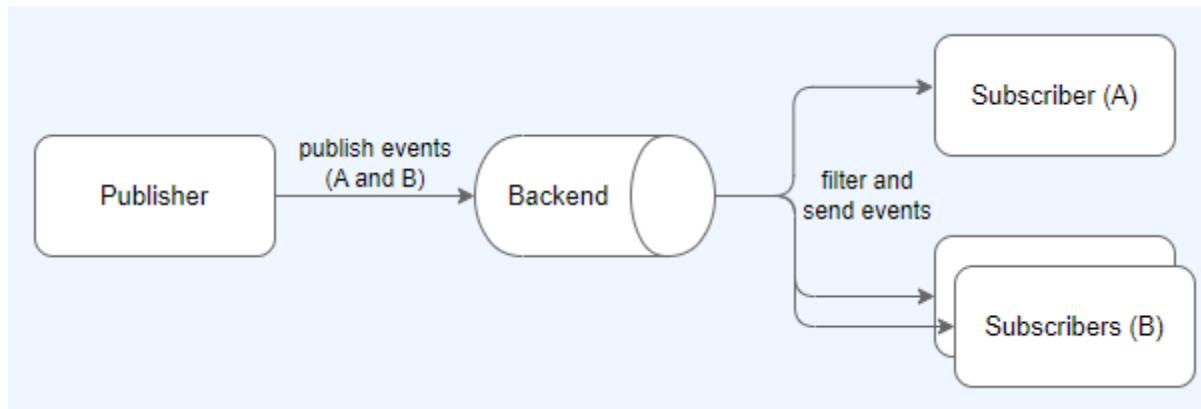
 Correct

Correct. Istio's components are installed in the istio-system namespace and the APIRule custom resource is translated into Istio components (VirtualService).

# Discovering Kyma

- Kyma is an open-source project
- Makes it easier to develop and run applications on Kubernetes
- Features provided by Kyma (extending functionality of k8s)
  - Serverless
  - Eventing
  - Observability
  - Service Integrations
  - Service Mesh
  - API Gateway

# Eventing in Kyma

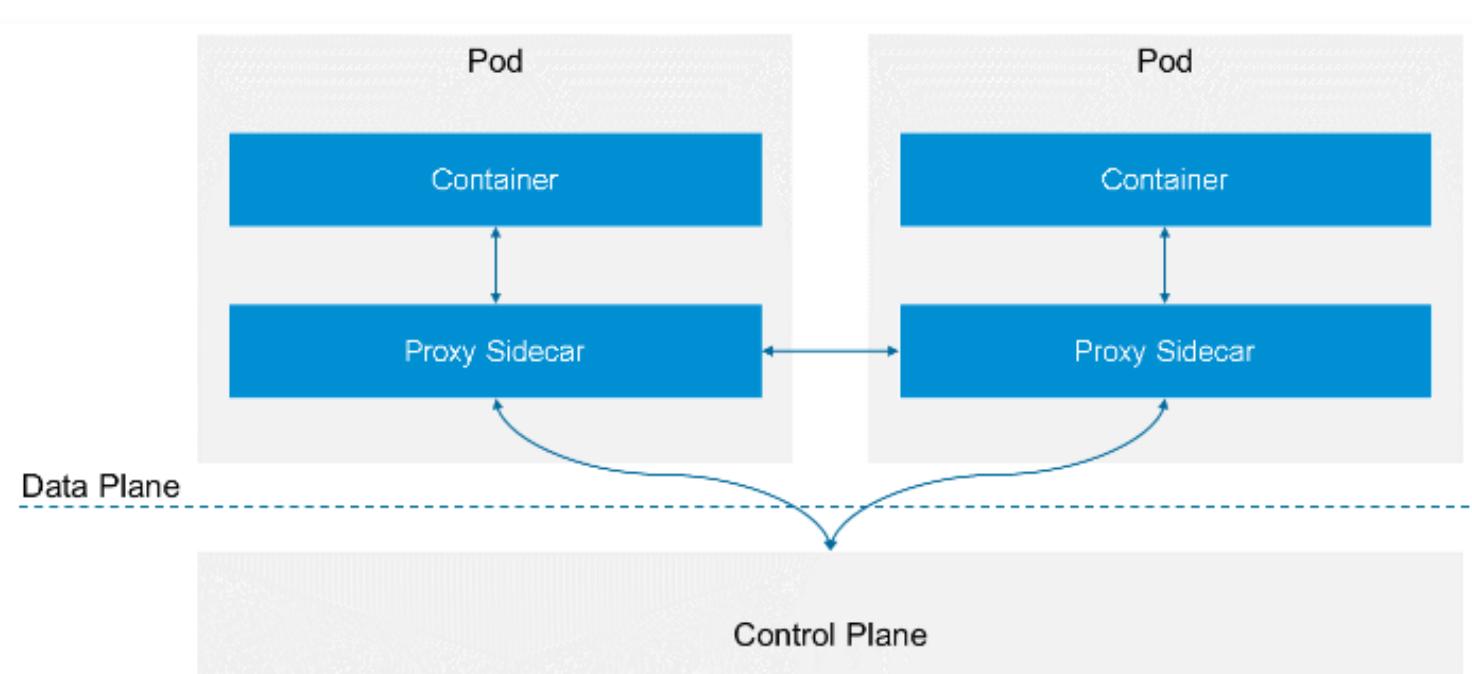
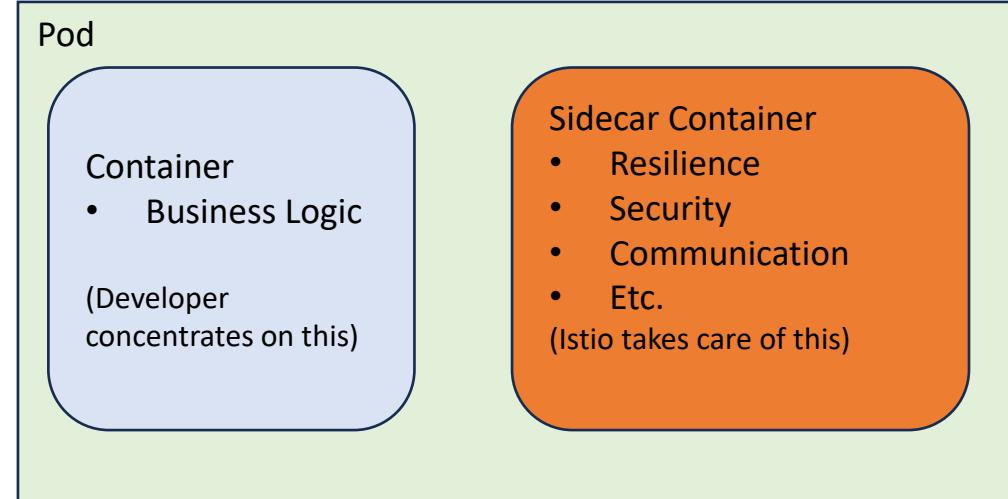


Eventing in Kyma – how it works ?

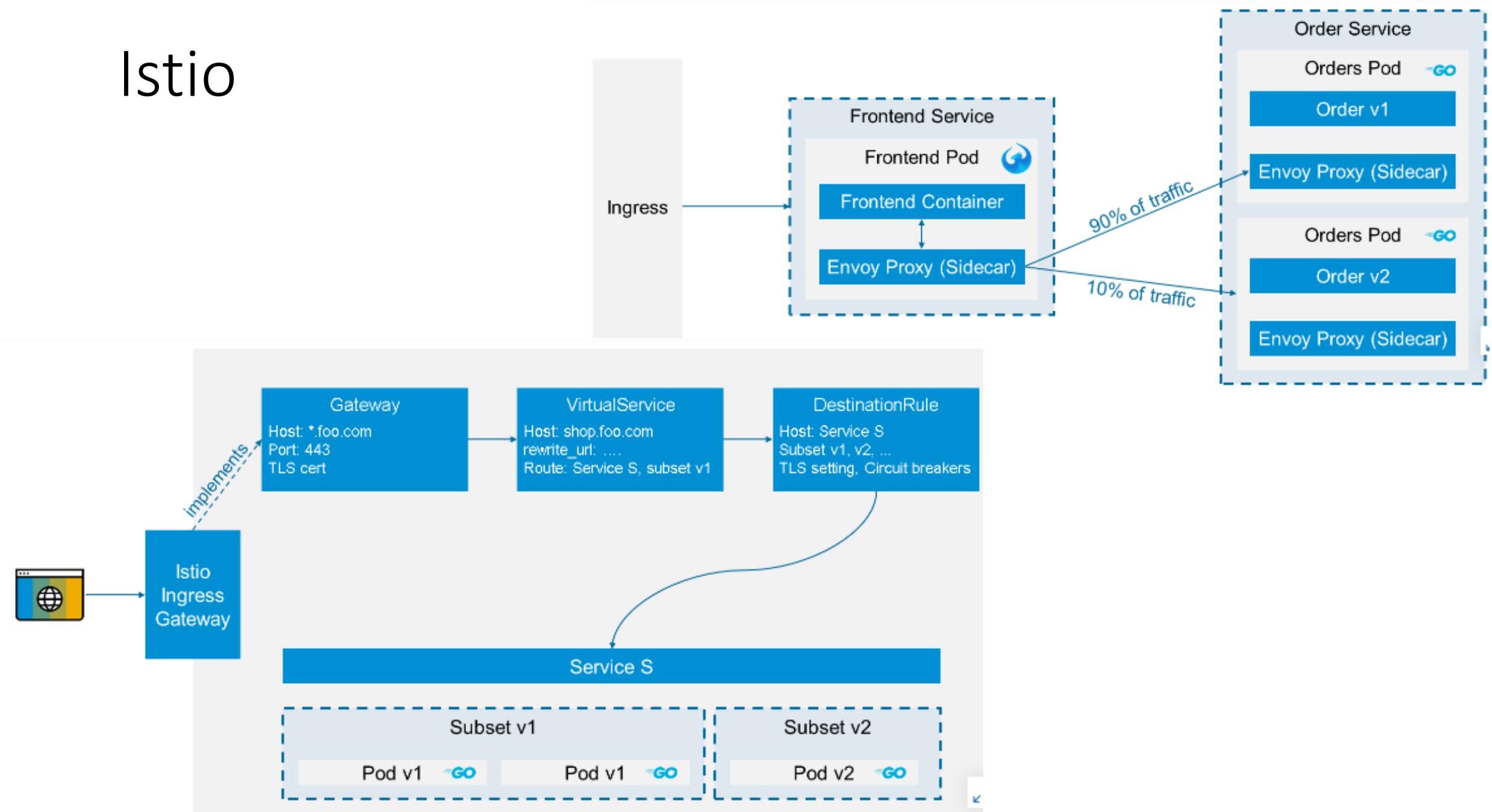
- Offer an HTTP endpoint, for example a function to receive events
- Specify the events the user is interested in using Kyma [Subscription CR](#)
- Send [CloudEvents](#) to the following HTTP endpoints on our Eventing Publisher Proxy Service
  - /publish for [CloudEvents](#)

# Discovering Service Mesh

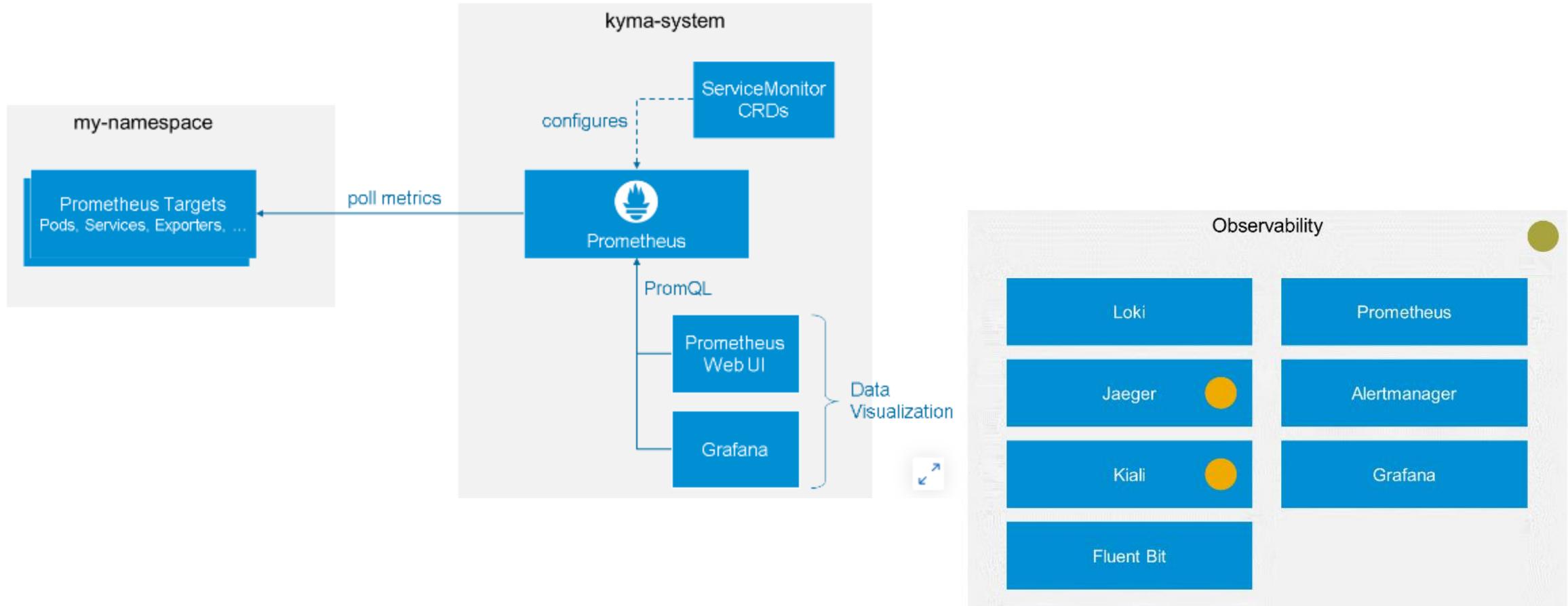
- Benefits of Service Mesh
  - Security
  - Observability
  - Traffic Management
  - Resilience
- Service Mesh Architecture



# Istio



# Observability and Monitoring



Not available in SAP BTP, Kyma Runtime

Limited in SAP BTP, Kyma Runtime

# StatefulSet

In most cases, pods are stateless

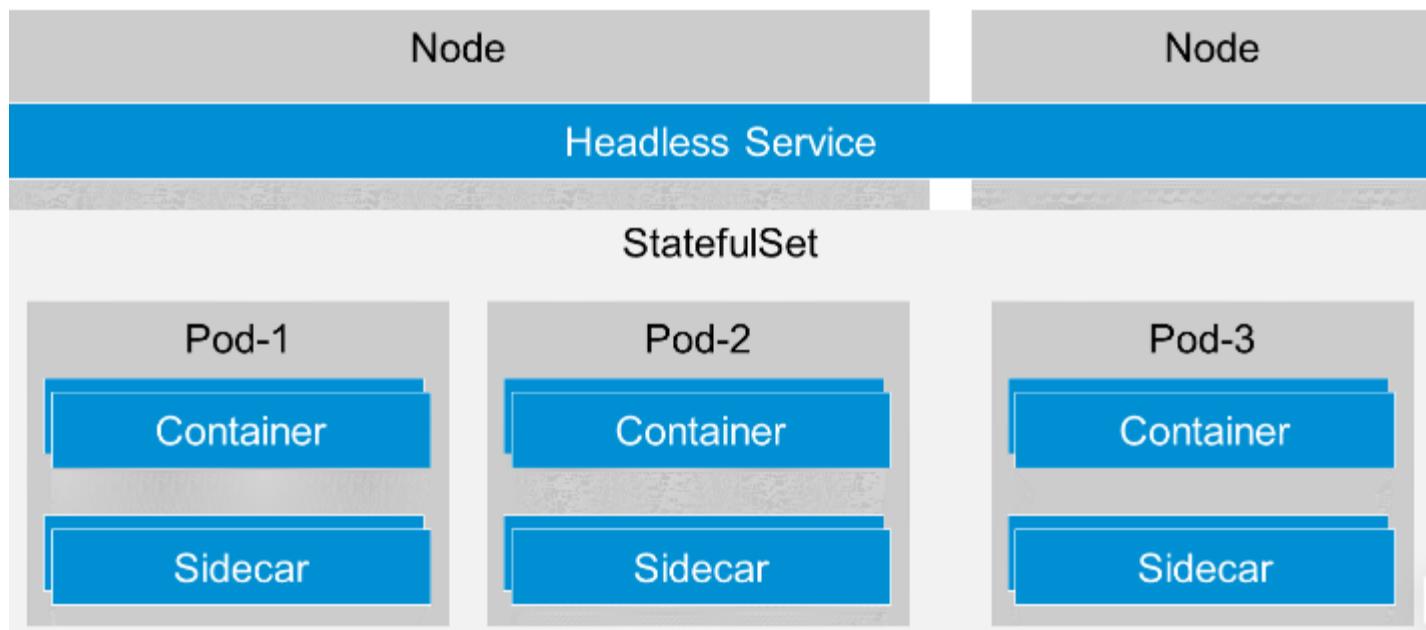
They die and are replaced **by any other pod with some random name**

But stateful applications (Database) need a unique, consistent identifier.

Use StatefulSets when you have the following requirements (e.g. Database)

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

When creating a StatefulSet, pods are created in sequential order (not any random order)



# StatefulSet

**Q4.** Which workload type is suitable for a stateful workload, such as a database?

A DaemonSet

B Deployment

C StatefulSet

D Pod

E ReplicaSet

F Correct

Correct. The workload type StatefulSet is suitable for a stateful workload.

# StatefulSet

**Q5.** Which workload type is most suitable for a stateless workload, such as a common microservice?

A DaemonSet

Deployment

C ReplicaSet

D StatefulSet

E Pod



Correct

Correct. The Deployment workload type is most suitable for a stateless workload.

# BTP Service Management

Service Management allows you to connect to SAP BTP services to your cluster

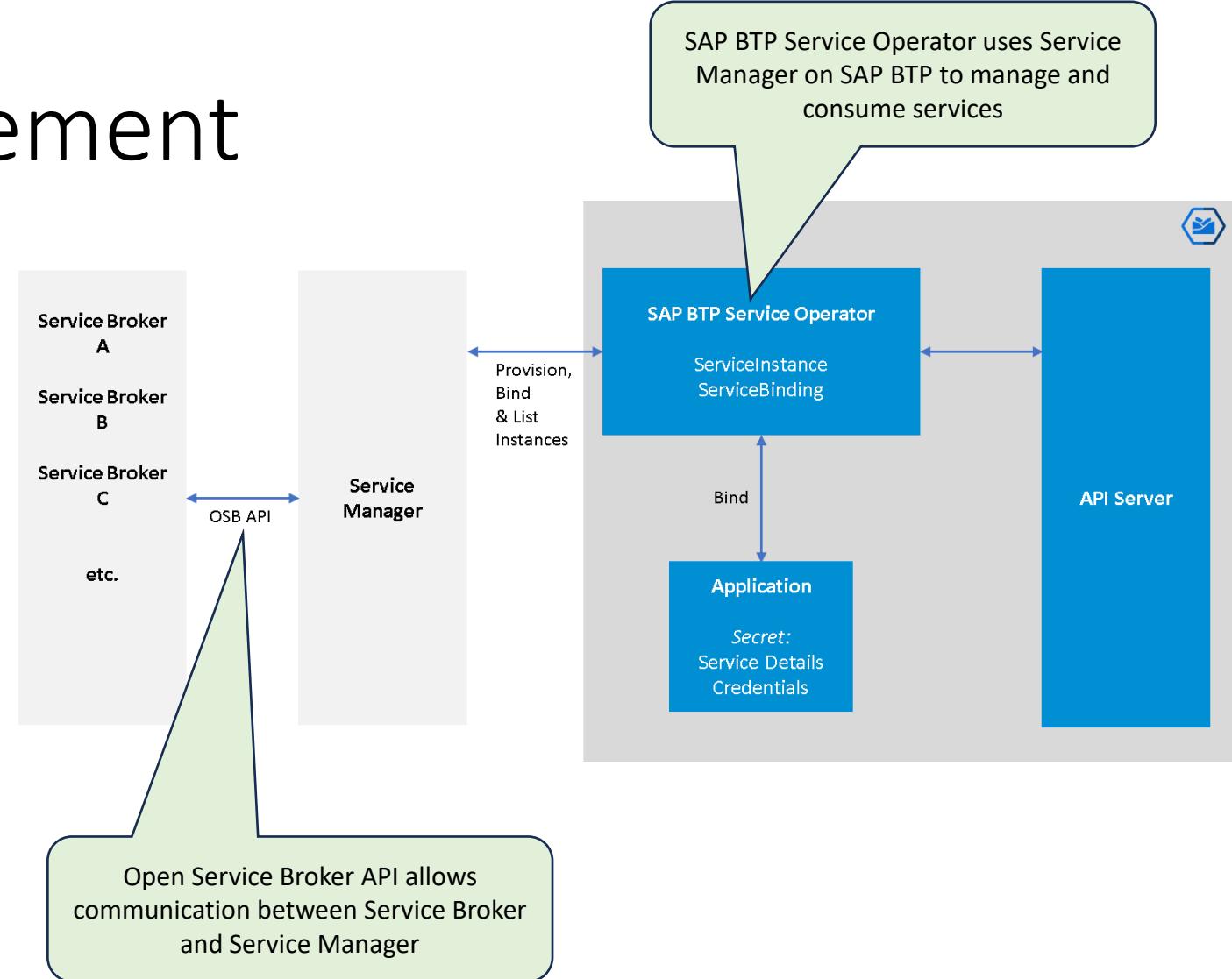
**Q2.** Which API comes into play for provisioning services on SAP BTP through SAP BTP, Kyma runtime?

- A Composite API
- B Queue Broker API
- C Messaging
- D Open Service Broker API



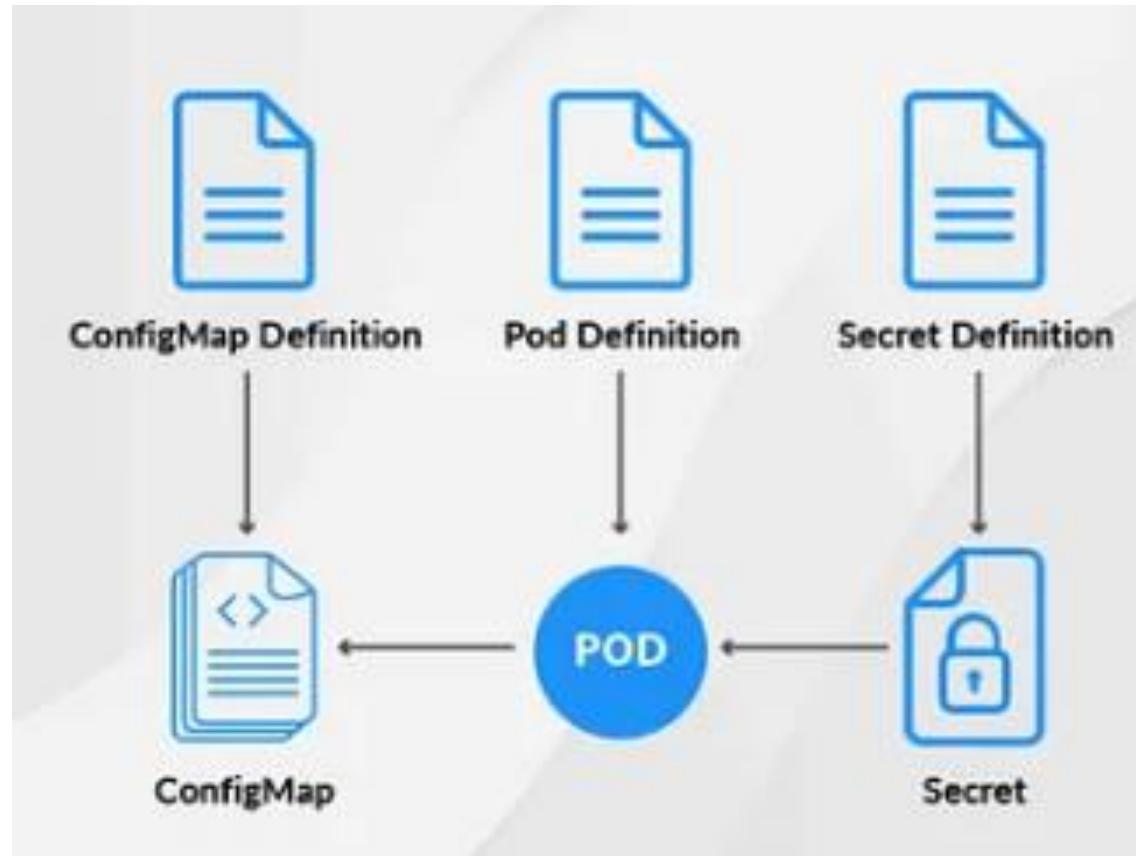
Correct

Correct. Open Service Broker API comes into play for provisioning services on SAP BTP through SAP BTP, Kyma runtime.



# ConfigMaps and Secrets

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: name-greeting
5 data:
6   GREETING: "Hello"
7   FIRSTNAME: "Kyma"
```



K8s has 2 native resources to store configuration data decoupled from application

## ConfigMaps

Non Sensitive data

## Secrets

Sensitive data (passwords, apikey etc.)

# Secrets

## Using Secrets

A Secret can be created as follows:

Code snippet

 Copy code

```
1  apiVersion: v1
2  kind: Secret ←
3  metadata:
4    name: my-secret
5  type: Opaque # default type different types available...
6  data:
7    username: "YWRtaW4=" # This is the base64 encoded string "admin"
8    password: "dG9wc2VjcmV0" # This is the base64 encoded string "topsecret"
```

[Collapse](#)

The above listed manifest creates a Secret with two keys: `username` and `password`. Kubernetes also comes with different [Secret types](#)  that can be used to store different types of sensitive data. For example, you can use the `kubernetes.io/dockerconfigjson` type to store a Docker config file. This is useful if you want to pull images from a private Docker registry.

# Secrets

If you want to pull the image from a private registry, you can do so by providing the credentials in the deployment manifest to `spec.template.spec`:

Code snippet

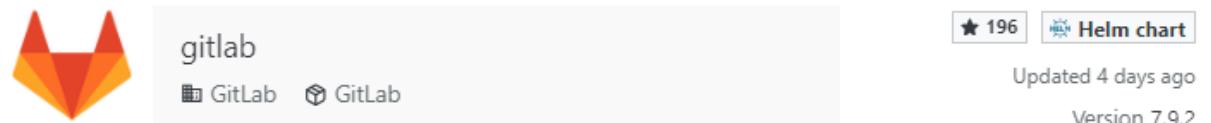
 Copy code

```
1  ...
2  spec:
3  ...
4  template:
5  ...
6  spec:
7    imagePullSecrets:
8      - name: myregistrykey
9    containers:
10      - name: hello-kyma
11        image: ghcr.io/sap-samples/kyma-runtime-learning-journey/hello-kyma:1.0.0
12        ports:
13          - containerPort: 8080
```

Collapse

# What is Helm ?

- Package manager for k8s
- Let's say you want to deploy gitlab
  - Might need the following (yaml files)...
    - Service to expose the endpoint
    - Persistent volume
    - Secret, ConfigMaps etc.
- Helm makes it super easy
  - *helm install gitlab*
  - *helm upgrade gitlab*
  - *helm uninstall gitlab*



GitLab is the most comprehensive AI-powered DevSecOps Platform.

Integration and delivery



## Q2. What is Helm?

A A tool for managing Kubernetes clusters.

✓ A tool for packaging and managing Kubernetes applications.

C A tool for restricting access to Kubernetes resources.

D A tool for managing Kubernetes users.

✓ Correct

Correct. Helm is a tool for packaging and managing Kubernetes applications.

# Service

In Kubernetes, a Service manifest file defines how to expose your application. While not all sections are mandatory, there are a few essential ones:

1. **apiVersion**: This field specifies the API version that your manifest is using. For example, `v1`.
2. **kind**: Defines the type of Kubernetes resource being described. For a Service manifest, this would be `Service`.
3. **metadata**: This section includes metadata about the Service, such as its name and labels.
4. **spec**: This is the most important section, as it defines the desired state of the Service. At a minimum, it should contain:
  - **selector**: Specifies the set of Pods targeted by this Service.
  - **ports**: Defines the ports that the Service will listen on.

```
yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

# Key Summary Points – Unit 6

Q1. In what ways can forms be used?

A For sending email

 For approving a sales order

 For sending notification to requestor

D For accessing automation



Correct

Correct. Forms can be used for approving a sales order and for sending notification to requestor.

[Order Processing Form](#)

[Approval Form](#)

[Order Confirmation Notification Form](#)

[Order Rejection Notification Form](#)

# Key Summary Points – Unit 6

Q8. What are the benefits of using forms?

 Quick, simple, and easy way to create interactive UI

 B Create documents for process

 One layout, one access and traceable action

 Streamline approvals in the business process

 E Send email templates to users

 Correct

Correct. One benefit of using forms is that it is a quick, simple and easy way to create interactive UI. Other benefits of using forms are that you only have one layout, one access and traceable action, and streamline approvals in the business process.

# Live Process Content

**Q2.** Which of the following steps need to be taken to activate Live Process Content from the SAP Build Process Automation Store?

Import the content

Browse the store

C Copy the content

D Export the content

Correct

Correct. If you want to activate Live Process Content from the SAP Build Process Automation Store, you have first to search for it in the SAP Build Process Automation Store and then import it.

# Live Process Content

**Q1.** Which of the following artifacts are usually part of an added Live Process from the SAP Build Process Automation Store?



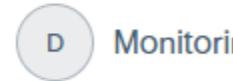
Decision



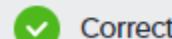
Visibility Scenario



Preconfigured APIs



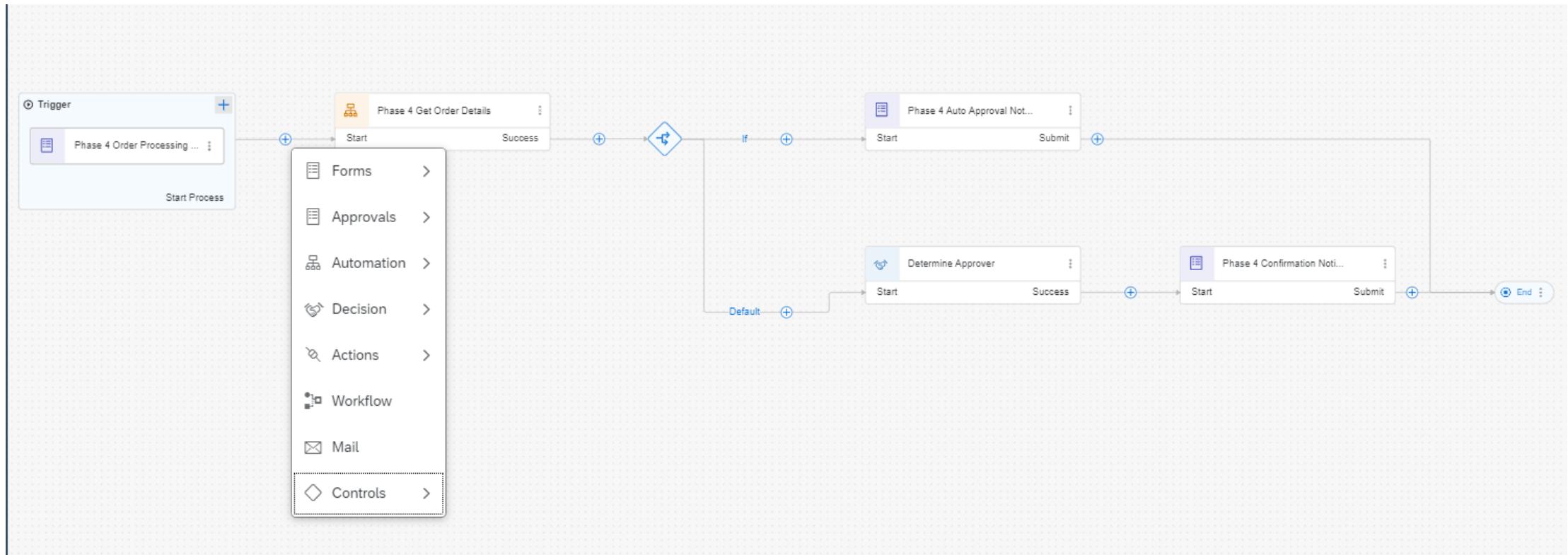
Monitoring Metrics



Correct

Correct. The Decision and Visibility Scenario are usually part of an added Live Process from the SAP Build Process Automation Store.

# Process Builder Artifacts



# Key Summary Points – Unit 6

Q6. Which configurations are necessary for Process Conditions?

 A Add a condition to the process

 B Map the condition input with process content

 C Configure If and else conditions

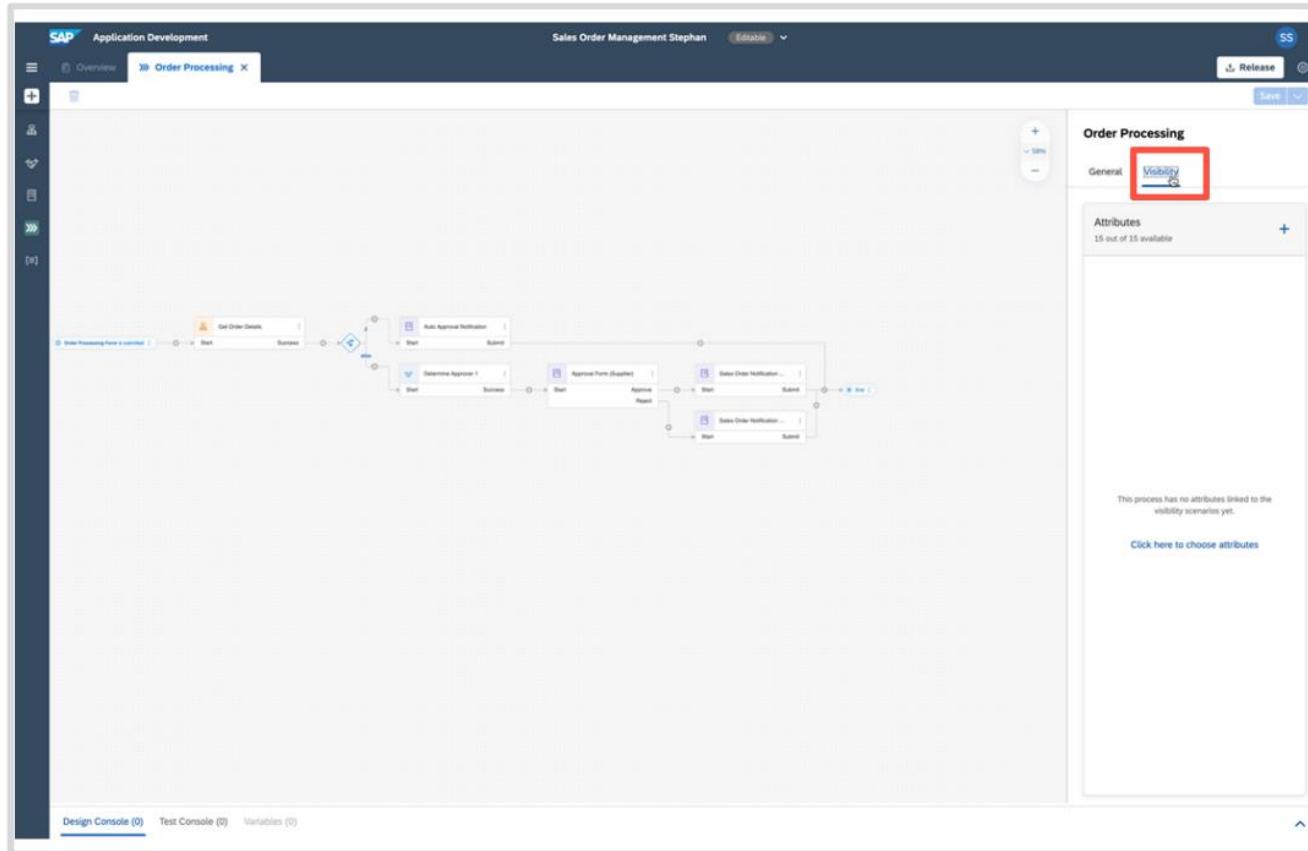
 D Define the process flow for different conditional paths

 E Add environment variables

 F Correct

Correct. The following configurations are necessary for Process Conditions: Add a condition to the process, Configure If and else conditions and Define the process flow for different conditional paths.

# Key Summary Points – Unit 6



## Prepare the Process to be Consumed

In the Process Builder open your process and on the right-hand side select the tab **Visibility**. Here you define the connection between the process context and the soon to be created visibility scenario.

Select the **Red Box** to **Continue**.

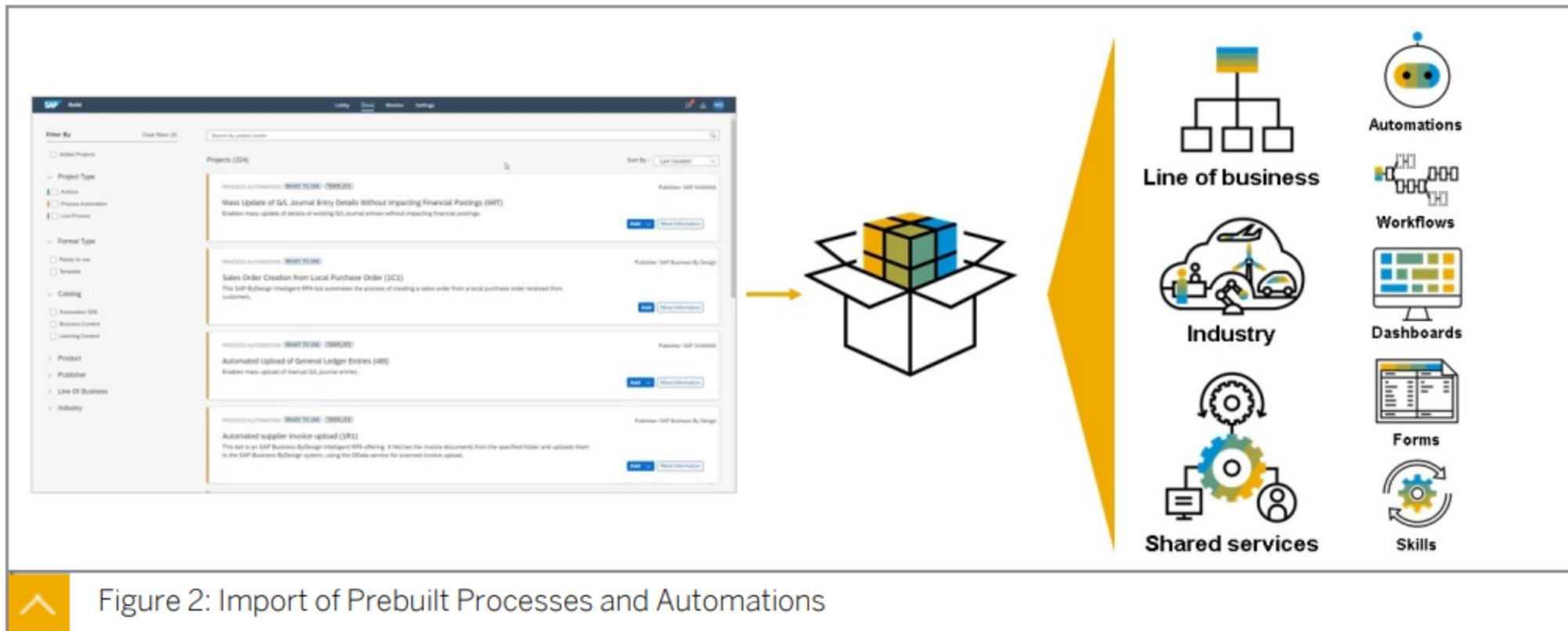
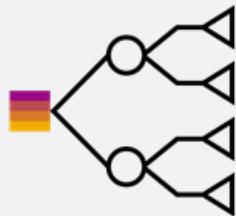


Figure 2: Import of Prebuilt Processes and Automations

With this option, you can make use of a wide variety of prebuilt, directly usable packages that are made available via the built-in Store. The packages span across lines of business, industries, and shared functions. The package content ranges from business processes to templated automations, from form templates to process visibility projects, as well as actions, also called skills. Also, there are complete packages containing all of the different artifacts. These packages are mostly ready to plug and play, only requiring minimal configuration efforts. The amount of content is continuously being extended.



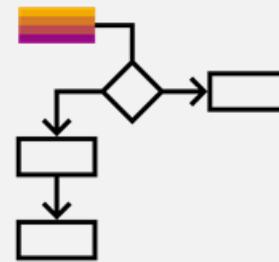
## Decision

Capability through which you can model and manage complex business logic in a well-formed structure that is easy to maintain.



## Policy/Policies

A decision consists of one or more policies. Each policy consists of a collection of rules.

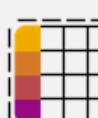


## Decision Diagram

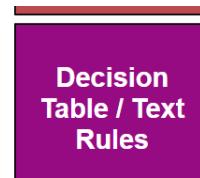
a flow chart that describes the execution flow of the decision logic from the input to the output.



Decision Table Rule



Text Rule



**Decision Table Rule** is a collection of input and output rule expressions in a tabular representation.  
**Text Rule** is a collection of rule expressions in a simple if-then format.

# Key Summary Points – Unit 3



Multiple event handlers can be registered for each event phase

- They are executed in the order in which they appear

Single event handler can also handle multiple events

All services are event emitters

- Events can be sent to them – READ, CREATE etc.
- Events can be emitted by them
- Register event handlers with services to react to any event
- `srv.on ("READ", Risks, (req, next) => {});`
- `srv.emit("customEvent", payload);`

# Key Summary Points – Unit 1, 2

`cds init <Name of Project>`

Folders created

- `app`
- `db`
- `srv`

Files created

- `package.json`

# Key Summary Points – Unit 3

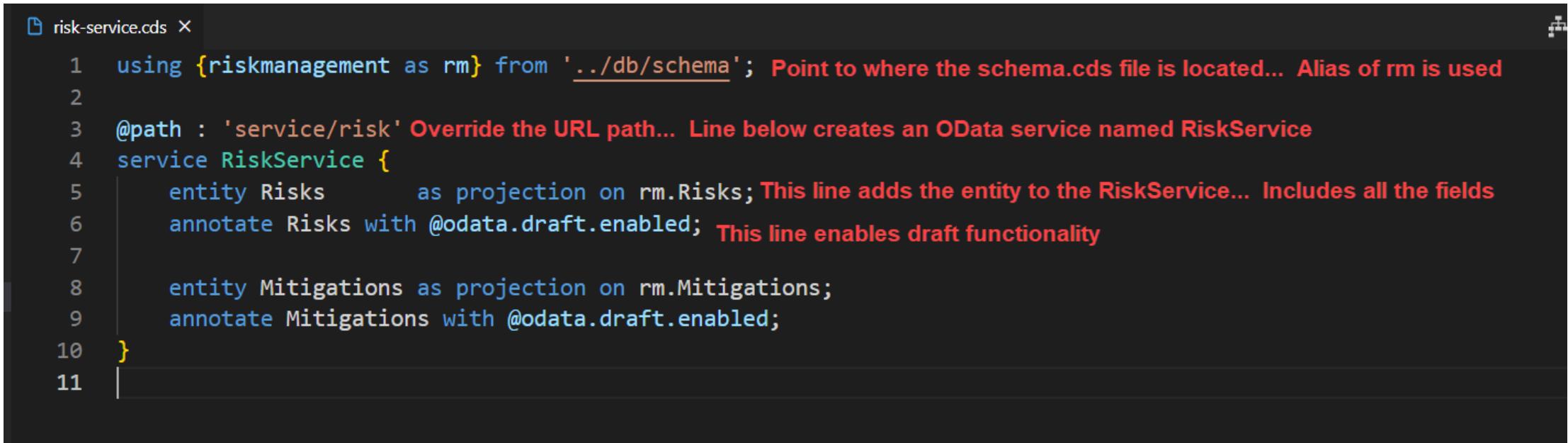
## **Recommended folders for various artifacts**

**app** – UI artifacts

**db** – database artifacts

**srv** – Service related artifacts

# Step 3 – Generic Handlers



```
risk-service.cds ×

1  using {riskmanagement as rm} from '../db/schema'; Point to where the schema.cds file is located... Alias of rm is used
2
3  @path : 'service/risk' Override the URL path... Line below creates an OData service named RiskService
4  service RiskService {
5    entity Risks      as projection on rm.Risks; This line adds the entity to the RiskService... Includes all the fields
6    annotate Risks with @odata.draft.enabled; This line enables draft functionality
7
8    entity Mitigations as projection on rm.Mitigations;
9    annotate Mitigations with @odata.draft.enabled;
10 }
11 |
```

A draft is a temporary version of a business entity that has not yet been explicitly saved as an active version.

Drafts are used:

- To keep unsaved changes if an editing activity is interrupted, allowing users to resume editing later.
- To prevent data loss if an app terminates unexpectedly.
- As a locking mechanism to prevent multiple users from editing the same object at the same time, and to make users aware when there are unsaved changes by another user.

# Key Summary Points – Unit 1, 2

## OData

- Data access protocol built on core protocols like HTTP and commonly accepted methodologies like REST
- Uses URI to address and access data feed resources
- Documents associated with each OData Service
  - Service Document
  - Service Metadata Document
- Supports 2 formats for representing resources
  - XML based AtomPub
  - JSON

# Key Summary Points – Unit 1, 2

## JSON

- Open standard file format and data interchange format
- Uses human-readable text to store and transmit data objects
- Consists of key-value pairs and arrays
- Based on JavaScript objects

## YAML

- Unicode based data serialization language
- YAML is a strict JSON superset – this means all valid JSON files are valid YAML files
- Support for serializing arbitrary native data structures

# Key Summary Points – Unit 5

## Question 4

*Choose the correct answer(s).*

Which statements about YAML files are correct? (Choose 2)

- YAML uses whitespace indentation for structuring purposes.
- YAML uses tab indentation for structuring purposes.
- YAML uses hyphens: - for comments.
- YAML uses hashes: # for comments.



## Feedback

Correct. The following statements are correct: "YAML uses whitespace indentation for structuring purposes", and "YAML uses hashes: # for comments".

# Key Summary Points – Unit 3

```
'  
this.after("READ", Risks, (data) => {  
  const risks = Array.isArray(data) ? data : [data];  
  
  risks.forEach((risk) => {  
    if (risk.impact >= 100000) {  
      risk.criticality = 1;  
    } else {  
      risk.criticality = 2;  
    }  
  });  
});
```

# Key Summary Points – Unit 3

The key takeaways for programming errors are:

- **Fail loudly:** Do not hide errors and continue silently. Ensure to log unexpected errors correctly. Don't catch errors you can't handle.
- Don't develop in a defensive fashion. Focus on your business logic and only handle errors when you know they will occur. Use try/catch blocks only when necessary.

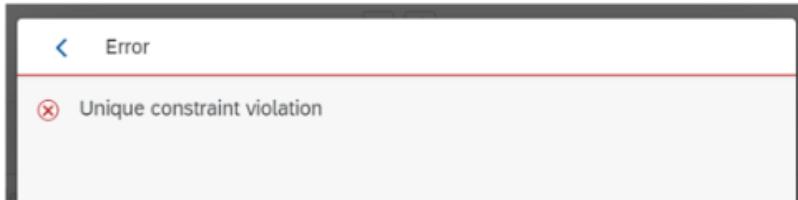
Never try to catch and handle unexpected errors, rejections of promises, and so on. If it is unexpected, you cannot handle it correctly. If you could, it would be expected (and should already be handled). Even if your apps should be stateless, you can never be 100% sure that a shared resource was not affected by the unexpected error. Therefore, you should never allow an app to continue running after such an event, especially for multi-tenant apps where there is a risk of information disclosure.

Following these guidelines will make your code shorter, clearer and simpler.

# Error Handlers

## Never Hide the Causes of Errors

When an error occurs, it should be possible to know the root cause. The CAP SDK for Node.js also throws exceptions, for example, when a CRUD operation violates the foreign key constraints. In this case, the framework throws the exception `UNIQUE_CONSTRAINT_VIOLATION`. The problem in this case is that the end user will only see a cryptic error message:



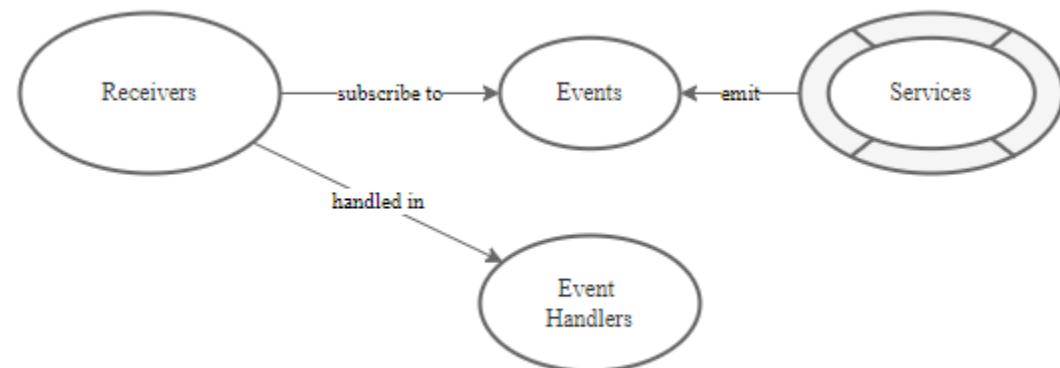
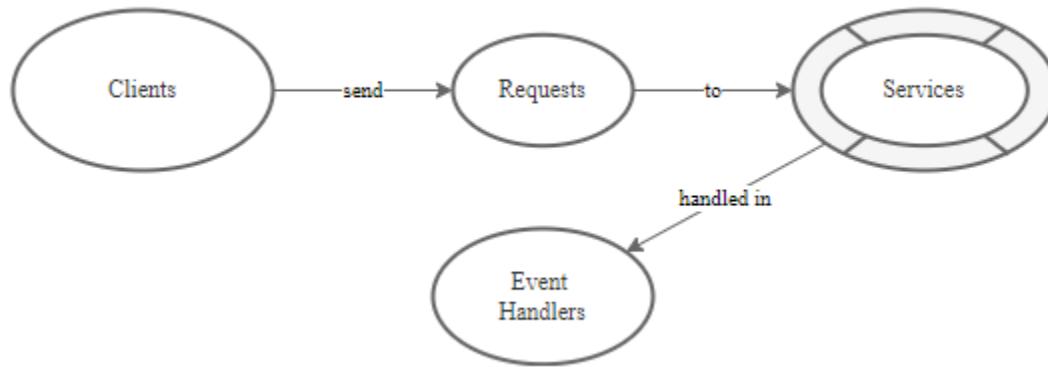
**Q8.** What can you do to provide meaningful error messages to users in your CAP application?

- A Hide the cause of errors
- B Catch exceptions
- C Register an error handler
- D Register an event handler



Correct  
Correct. To provide meaningful error messages to users in your CAP application, you can register an error handler.

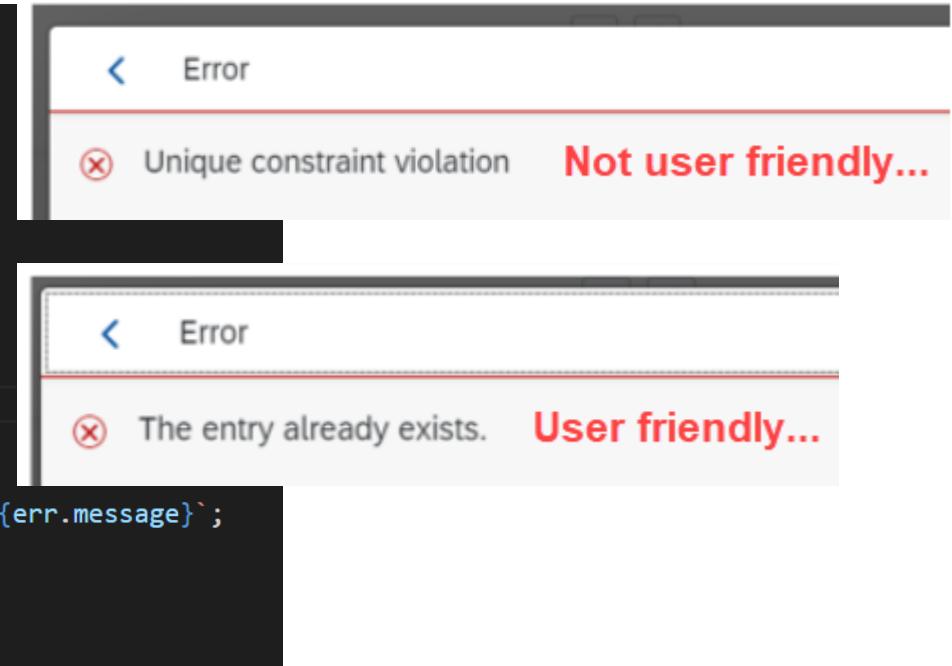
# Key Summary Points – Unit 3



# Error Handling

```
/*
 * Custom error handler
 *
 * throw a new error with: throw new Error('something bad happened');
 *
 */
this.on("error", (err, req) => {
  switch (err.message) {
    case "UNIQUE_CONSTRAINT_VIOLATION":
      err.message = `The entry already exists.`;
      break;

    default:
      err.message = `An error occurred. Please retry. Technical error message: ${err.message}`;
      break;
  }
});
```



Register an error handler in your service implementation

- Provide meaningful error message to end user

# Key Summary Points – Unit 4

- Cloud Foundry environment BTP Connectivity
  - Connectivity Service – **Connectivity proxy to access on-premise resource**
  - Destination Service – **Retrieve and store technical info about target resource**

I want to...	Services required
Connect to publicly available Services	Destination Service
Connect to On-Premise Services	Destination Service, Connectivity Service

# Key Summary Points – Unit 4

In SAP Fiori, drafts are used as follows:

- To keep unsaved changes if an editing activity is interrupted, allowing users to resume editing later.
- To prevent data loss if an app terminates unexpectedly.
- As a locking mechanism to prevent multiple users from editing the same object concurrently, and to make users aware when there are unsaved changes by another user.

# Key Summary Points – Unit 5

SAP BTP Cockpit

Subaccount: trial - Overview

General Cloud Foundry Environment Kyma Environment Entitlements

Entitlements: 79 Instances and Subscriptions: 30

Subdomain: d885dfc8trial Provider: Amazon Web Services (AWS)

Tenant ID: 2d557e89-9d54-4e15-8c47-302c0a23c46c Region: US East (VA)

Subaccount ID: 2d557e89-9d54-4e15-8c47-302c0a23c46c Environment: Multi-Environment

**Cloud Foundry Environment**

Org Name: d885dfc8trial Automatically created by System  
API Endpoint: <https://api.cf.us10.hana.ondemand.com>  
Org ID: 22362934-63c1-4090-b25c-57742a72af9

Manage environment instance

Disable Cloud Foundry Disable Cloud Foundry environment at Subaccount

Trial Home / d885dfc8trial / mysubaccount

Subaccount: mysubaccount - Overview

General Cloud Foundry Environment Entitlements

Entitlements: 23 Instances and Subscriptions: 2

Subdomain: mysubaccount-23y9yjjc Provider: Amazon Web Services (AWS)

Tenant ID: ccf09091-c94b-403d-b726-25b7eb0bc0e5 Region: US East (VA)

Subaccount ID: ccf09091-c94b-403d-b726-25b7eb0bc0e5 Environment: Multi-Environment

**Cloud Foundry Environment**

You are not currently using Cloud Foundry capabilities.

Enable Cloud Foundry Enable Cloud Foundry environment at Subaccount level

CF Org is automatically created by the system – when you enable Cloud Foundry environment  
Subaccount and Org have a 1:1 relationship

# Key Summary Points – Unit 5

## Question 8

*Choose the correct answer(s).*

Which tools can you use to manage the SAP BTP, Cloud Foundry environment?

- SAP Business Application Studio
- Eclipse
- CF CLI
- SAP BTP cockpit

## Feedback



Correct. You can use the following tools to manage the SAP BTP, Cloud Foundry environment: CF CLI and SAP BTP cockpit.

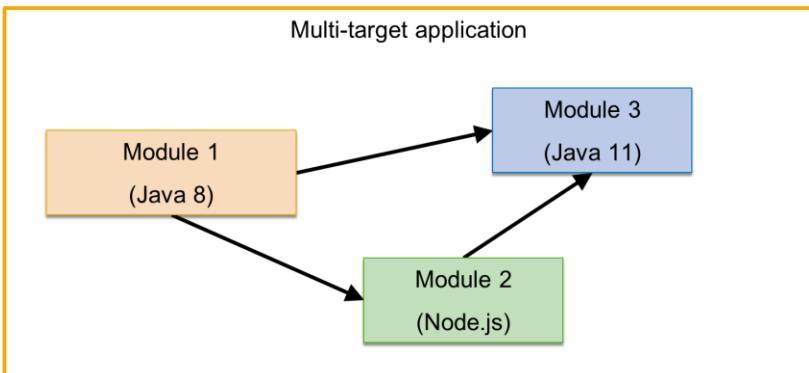
# Key Summary Points – Unit 6

```
service RiskService {  
    entity Risks @restrict : [  
        {  
            grant : ['READ'],  
            to : ['RiskViewer'] RiskViewer Role - Can only READ  
        },  
        {  
            grant : ['*'],  
            to : ['RiskManager'] RiskManager Role - Can do everything  
        }  
    ]) as projection on rm.Risks;  
  
    annotate Risks with @odata.draft.enabled;  
    entity Mitigations @restrict : [  
        {  
            grant : ['READ'],  
            to : ['RiskViewer'] RiskViewer Role - Can only READ  
        },  
        {  
            grant : ['*'],  
            to : ['RiskManager'] RiskManager Role - Can do everything  
        }  
    ]) as projection on rm.Mitigations;  
  
    annotate Mitigations with @odata.draft.enabled;
```

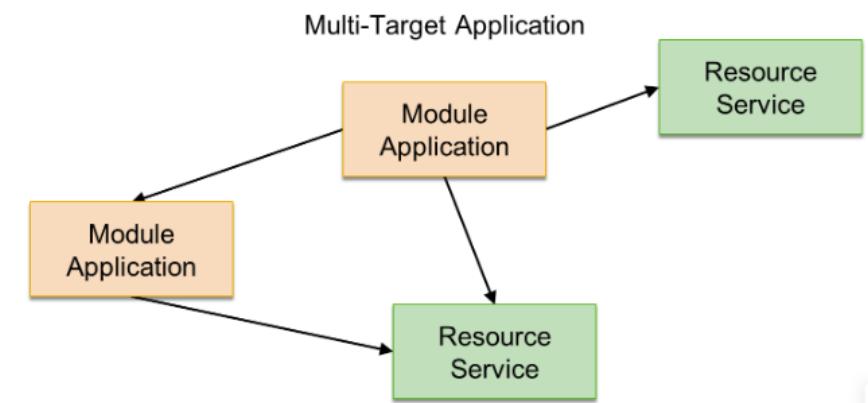
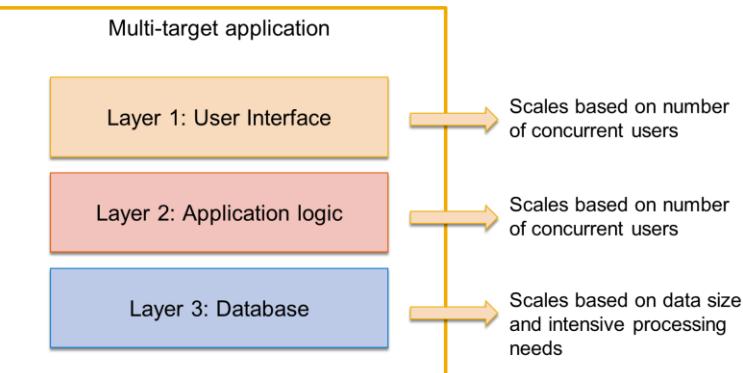
[srv/risk-service.cds](#)

# Multi Target Applications

Use MTA to integrate different programming languages or runtime environments



Example: Use of 3 tiers architecture in the application, to have better application scalability



**Q1.** Which tool do you use to build an MTA?

mbt

multiapps

Maven

Npm

Correct

This is correct. You build a MTA using the mbt.

# Key Summary Points – Unit 5

## Question 2

*Choose the correct answer(s).*



What are advantages of using an MTA file for deployment? (Choose 2)

- It supports red - green deployment.
- It supports blue-green deployment.
- It provides workflows.
- It provides a build tool.

## Feedback

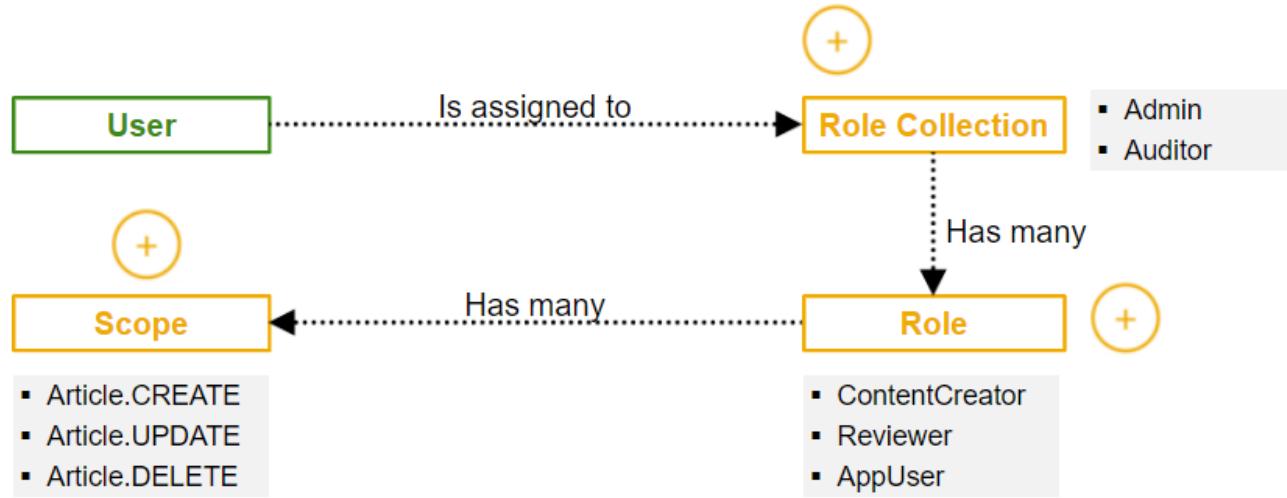
Correct. The advantages of using an MTA file for deployment are: "it supports blue-green deployment", and "it provides a build tool".

```
mbt build -t ./  
mbt build
```

# Key Summary Points – Unit 6

```
1 "scopes": [
  {
    "name": "$XSAPPNAME.RiskViewer",
    "description": "RiskViewer"
  },
  {
    "name": "$XSAPPNAME.RiskManager",
    "description": "RiskManager"
  }
],
"attributes": [],
"role-templates": [
  {
    "name": "RiskViewer",
    "description": "generated",
    "scope-references": [
      "$XSAPPNAME.RiskViewer"
    ],
    "attribute-references": []
  },
  {
    "name": "RiskManager",
    "description": "generated",
    "scope-references": [
      "$XSAPPNAME.RiskManager"
    ],
    "attribute-references": []
  }
]
```

# Key Summary Points – Unit 6



- In SAP BTP, CF environment, a single authorization is called **Scope**
- **Scopes** cannot be assigned to users directly – They are packaged into Roles
- **Scopes** are prefixed with xsappname to make them uniquely identifiable
- **Role** has many Scopes
- **Role-Collections** contain 1 or more Roles
- **Role-Collections** can be assigned to a User

# Key Summary Points – Unit 7

The screenshot shows the SAP BTP Cockpit interface. The left sidebar contains navigation links for Overview, Services, Instances and Subscriptions, Cloud Foundry, HTML5 Applications, Connectivity, Security, Entitlements, and Usage Analytics. The main content area displays the 'Subaccount: trial - Instances and Subscriptions' page, which includes a search bar and filters for All Services, All Plans, and All Statuses. A callout box provides instructions to manage service instances via Cloud Foundry - Spaces. Below this, tabs for Subscriptions (9), Instances (20), and Environments (1) are shown, with 'Subscriptions (9)' being active. A table lists the applications to which the subaccount is subscribed, including their names, plans, creation dates, and last change dates, along with status indicators (e.g., Unsubscription Failed, Subscribed).

Application	Plan	Created On	Changed On	Status	Action
Product List MTA	default	17 Jan 2022	10 Apr 2022	Unsubscription Failed	...
Continuous Integration & Delivery	trial	10 Apr 2022	10 Apr 2022	Subscribed	...
Document Information Extraction Trial UI	default	26 Jan 2022	26 Jan 2022	Subscribed	...
Integration Suite	trial	17 Feb 2022	17 Feb 2022	Subscribed	...
SAP Business Application Studio	trial	17 Nov 2021	17 Nov 2021	Subscribed	...
Launchpad Service	standard	6 Jan 2022	24 Feb 2022	Subscribed	...
Web Analytics	standard	17 Nov 2021	17 Nov 2021	Subscribed	...
Subscription Management Dashboard	application	17 Jan 2022	17 Jan 2022	Subscribed	...
Workflow Management	saas-application	6 Jan 2022	6 Jan 2022	Subscribed	...

# Key Summary Points – Unit 6

The screenshot shows the SAP BTP Cockpit interface. On the left, the sidebar navigation includes: Overview, Services (with Service Marketplace selected), Cloud Foundry, HTML5 Applications, Connectivity (Destinations, Cloud Connectors), Security (selected), and Users (selected). The main content area displays the 'Subaccount: trial - Users' page, which lists a single user: milton.chandradas@sap.com. The right side shows the 'Role Collections' page, listing several collections with one highlighted: 'CICD Service Administrator'.

**Subaccount: trial - Users**

User Name	Identity Provider	Last Name	First Name	E-Mail	Last Updated	Last Logon	Actions
milton.chandradas@sap.com	Default identity provider	Chandra das	Milton	milton.chandradas@sap.com	17 Nov 2021, 23:31:55 (GMT-05:00)	24 Apr 2022, 02:10:02 (GMT-04:00)	<span>Delete</span> <span>&gt;</span>

**Role Collections**

Name	Description	Action
Business_Application_Studio_Developer	Allows developers to load and develop applications using SAP Business Application Studio.	<span>Delete</span> <span>&gt;</span>
<b>CICD Service Administrator</b>	Administrator Role Collection for the Continuous Integration & Delivery Service.	<span>Delete</span> <span>&gt;</span>
Document_Information_Extraction_UI_Admin_User_trial	A role collection containing the Document_Information_Extraction_UI_Admin_User_trial Role Template.	<span>Delete</span> <span>&gt;</span>
Integration_Provisioner_Auto	For provisioning and basic configuration of the Integration capabilities	<span>Delete</span> <span>&gt;</span>
ProductListViewer	Product List Viewer	<span>Delete</span> <span>&gt;</span>
RiskManager-dev	Manage Risks	<span>Delete</span> <span>&gt;</span>
SAP Web Analytics Customer Admin	Customer Admin	<span>Delete</span> <span>&gt;</span>
Subaccount Administrator	Administrative access to the subaccount	<span>Delete</span> <span>&gt;</span>

# CI CD

The image shows the SAP Continuous Integration and Delivery (CI/CD) interface. On the left, there is a list of repositories under the 'Repositories' tab. On the right, a modal dialog titled 'Add Repository' is open, showing fields for 'General Information' and 'Webhook Event Receiver'.

**SAP Continuous Integration and Delivery**

**Repositories (5)**

1

Name	URL
capauth-test-rt	<a href="https://github.com/richardtanha/capauth.git">https://github.com/richardtanha/capauth.git</a>
capcertification_repository	<a href="https://github.com/miltonchandradas/capcertification.git">https://github.com/miltonchandradas/capcertification.git</a>
mytest_repository	<a href="https://github.com/miltonchandradas/mytest.git">https://github.com/miltonchandradas/mytest.git</a>
productionbranch	<a href="https://github.com/miltonchandradas/rapdemo.git">https://github.com/miltonchandradas/rapdemo.git</a>
risk-management-repository	<a href="https://github.com/miltonchandradas/risk-management.git">https://github.com/miltonchandradas/risk-management.git</a>

2

3

**Add Repository**

**General Information**

Name: \*

Clone URL: \*

Credentials:

Cloud Connector:

**Webhook Event Receiver**

Type: GitHub

Webhook Credential: \*

State:

# Event handlers

## 1 .js file

.js file with same name as .cds file

```
// cat-service.cds
service CatalogService {...}

// cat-service.js
module.exports = (srv)=>{...}
```

## 2 .js file and @impl

.js file and annotation in .cds file

```
@impl: 'my-service.js'
service CatalogService {...}

// my-service.js
module.exports = (srv)=>{...}
```

## 3 cds.serve().with(...)

.js file or inline function passed to serve.with()

```
cds.serve('./cat-service').with('./cat-service.js')
// or
cds.serve('./cat-service').with (srv=> srv.on (...))
```

## 4 cds.s#erve() ...

Inline function passed to result of cds.serve()

```
const {CatalogService} = await cds.serve('./cat-service')

CatalogService.on ('READ','Books', req => {...})
// or
CatalogService.impl (srv=> srv.on (...))
```

## 5 cds.connect() ...

Inline function passed to result of cds.connect()

```
const {ExternalService} = await cds.connect('external-service') ExternalService.on ('some-event',
evt => {...})
```

# Key Summary Points – Unit 6

## Question 8

*Choose the correct answer(s).*

What does the Extended Services - User Account and Authentication (XSUAA) service enable your app to do?

- Store "real" users.
- Identify users by address and social security ID.
- Identify users by e-mail, userId, first and last name.
- Check users' roles to allow or prohibit actions.

## Feedback

Correct. XSUAA enables your app to identify users by e-mail, userId, first and last name and check users' roles to allow or prohibit actions.

# Key Summary Points – Unit 7

## Question 1

*Choose the correct answer(s).*



Which of the following statements about a GitHub Repository are correct?

- Anyone on the internet can see a public repository.
- Anyone on the internet can commit into a public repository.
- You choose who can see your private repository.
- You choose who can commit into your private repository.

## Feedback

Correct. Anyone on the internet can see a public repository. You choose who can see your private repository. You choose who can commit into your private repository.

# Key Summary Points – Unit 6

## [\*\*xs-security.json\*\* – Application Security Descriptor](#)

File that defines the details of authentication method and authorization types to use for access to your application

The contents of the `xs-security.json` are used to configure the OAuth 2.0 client; the configuration is shared by all components of an SAP multi-target application. The contents of the `xs-security.json` file cover the following areas:

- Authorization scopes  
A list of limitations regarding privileges and permissions and the areas to which they apply
- Attributes  
A list of as-yet undefined information or sources (for example, the name of a cost center)
- Role templates  
A description of one or more roles to apply to a user and any attributes that apply to the roles

# Key Summary Points – Unit 7

The screenshot shows the GitHub developer settings page. A red box highlights the 'Settings / Developer settings' link at the top left. On the left sidebar, a red box highlights the 'Personal access tokens' option under the 'Developer tools' section. The main area is titled 'Personal access tokens' and contains a single token entry:

git: https://github.com/ on AGSN34426591A at 04-Aug-2019 22:41 — gist, repo, workflow	Last used within the last 3 months	Delete
⚠ This token has no expiration date.		

A red box highlights the 'Generate new token' button at the top right of the token list. Another red box highlights the 'Revoke all' button at the far right.

Personal Access Tokens are an **alternative to using passwords** for authentication to GitHub when using the GitHub API or command line

GitHub automatically removes Personal Access Tokens that haven't been used in a year

# Key Summary Points – Unit 7

## Question 4

*Choose the correct answer.*

What does the source code management system use to trigger the CI server?

- Webhooks
- Web services
- HTTP PUT requests

## Feedback

Correct. The source code management system uses Webhooks to trigger the CI server.

# Key Summary Points – Unit 7

## Question 10

*Choose the correct answer.*

What is a "main line" in a source control management system used for?

- To automate deployment.
- To enable a reporting line for the project manager.
- To make developers' contribution transparent and avoid clashes.

## Question 11

*Determine whether this statement is true or false.*

A main line in a source control management system can contain feature branches.

- True
- False

# Key Summary Points – Unit 7

## Question 13

*Choose the correct answer.*

What do you use to retrieve the information about a change on the repository?

- A change document
- A webhook
- A PUT request to GitHub

## Question 15

*Choose the correct answer.*

What kind of request does the webhook send?

-   GET
- PUT
- POST

# Key Summary Points – Unit 7

The screenshot displays the SAP Continuous Integration and Delivery (CI/CD) interface. It is divided into two main sections: 'Jobs' and 'Builds'.

**Jobs Section:** This section shows a list of jobs associated with a repository. A red box highlights the 'Jobs (1)' tab in the top navigation bar. Below it, there are tabs for 'Repositories (1)' and 'Credentials (3)'. A search bar and a '+' button are also present. The table lists one job:

Name	State	Branch	Timed Triggers	Pipeline
risk-management-repo	ON	master	0	SAP Cloud Application Programming Model 1.0 >

A red box highlights the pipeline name 'SAP Cloud Application Programming Model 1.0' and the expand arrow icon to its right.

**Builds Section:** This section shows the build history for the selected job. A red box highlights the 'Builds' tab in the top navigation bar. Below it, there are tabs for 'General Information', 'Build Retention', 'Stages', and 'Timed Triggers'. A large red box highlights the 'Manually trigger build' button. The build history is listed in a table:

#	Commit	Time	Action
#3	Commit 5d5a0b3	6 min 22 · Apr 22, 2022, 6:52:39 PM	<b>Check build steps for errors</b> Retrigger Delete
#2	Commit ab69e92	10 min 06 · Apr 21, 2022, 10:47:52 PM	Retrigger Delete

Red boxes highlight the commit details and the 'Retrigger' and 'Delete' buttons for each build entry.

## INSTALLATION & GUI'S

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

### GitHub for Windows

<https://windows.github.com>

### GitHub for Mac

<https://mac.github.com>

For Linux and Solaris platforms, the latest release is available on the official Git web site.

### Git for All Platforms

<http://git-scm.com>

## SETUP

Configuring user information used across all local repositories

**git config --global user.name "[firstname lastname]"**

set a name that is identifiable for credit when reviewing version history

**git config --global user.email "[valid-email]"**

set an email address that will be associated with each history marker

**git config --global color.ui auto**

set automatic command line coloring for Git for easy reviewing

## SETUP & INIT

Configuring user information, initializing and cloning repositories

**git init**

initialize an existing directory as a Git repository

**git clone [url]**

retrieve an entire repository from a hosted location via URL

## STAGE & SNAPSHOT

Working with snapshots and the Git staging area

**git status**

show modified files in working directory, staged for your next commit

**git add [file]**

add a file as it looks now to your next commit (stage)

**git reset [file]**

unstage a file while retaining the changes in working directory

**git diff**

diff of what is changed but not staged

**git diff --staged**

diff of what is staged but not yet committed

**git commit -m "[descriptive message]"**

commit your staged content as a new commit snapshot

## BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

**git branch**

list your branches. a \* will appear next to the currently active branch

**git branch [branch-name]**

create a new branch at the current commit

**git checkout**

switch to another branch and check it out into your working directory

**git merge [branch]**

merge the specified branch's history into the current one

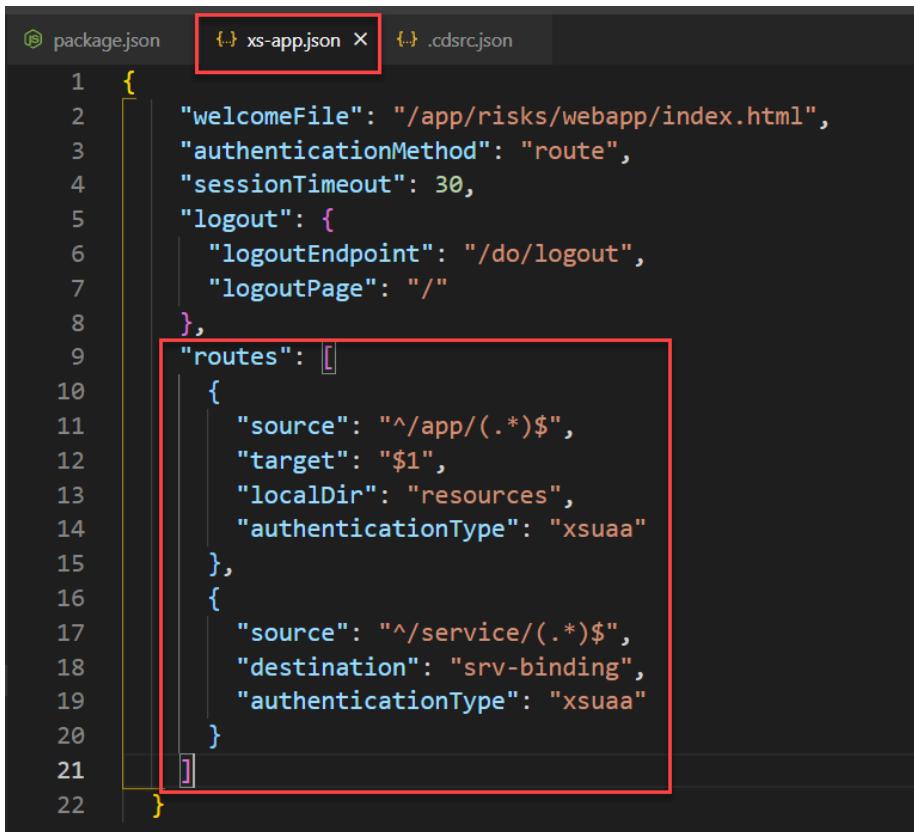
**git log**

show all commits in the current branch's history

# Key Summary Points – Unit 6

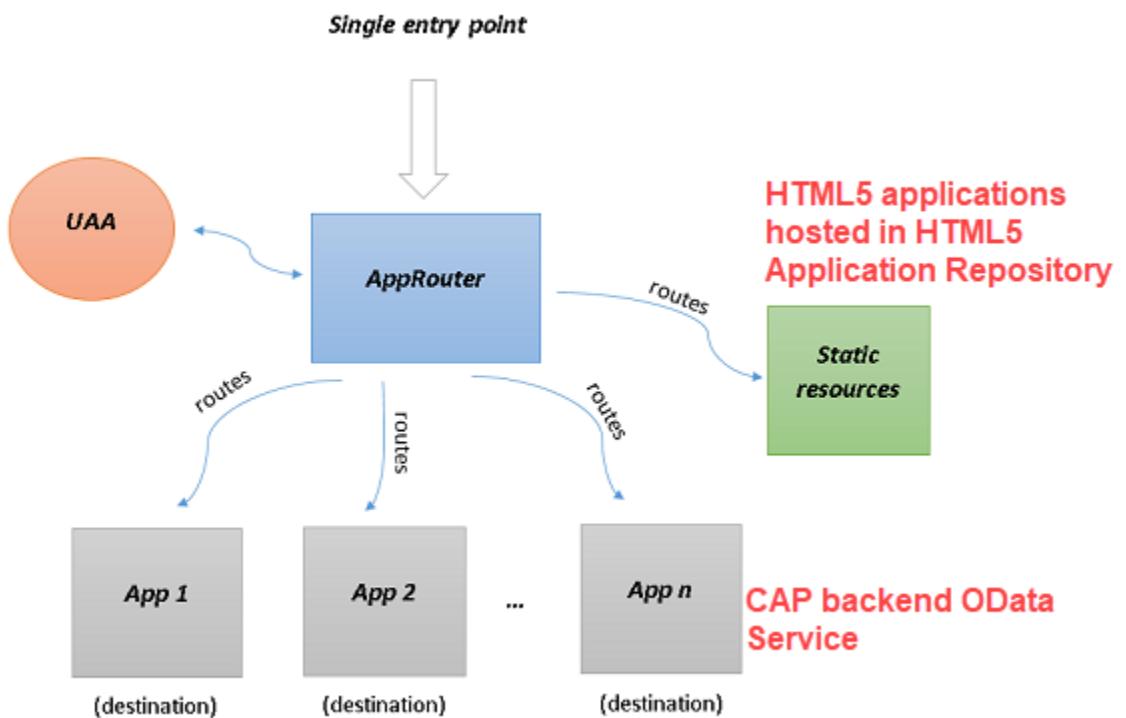
## xs-app.json – Application Router Configuration

File contains configuration information used by the application router



```
1  {
2    "welcomeFile": "/app/risks/webapp/index.html",
3    "authenticationMethod": "route",
4    "sessionTimeout": 30,
5    "logout": {
6      "logoutEndpoint": "/do/logout",
7      "logoutPage": "/"
8    },
9    "routes": [
10      {
11        "source": "^/app/(.*)$",
12        "target": "$1",
13        "localDir": "resources",
14        "authenticationType": "xsuaa"
15      },
16      {
17        "source": "^/service/(.*)$",
18        "destination": "srv-binding",
19        "authenticationType": "xsuaa"
20      }
21    ]
22 }
```

# Key Summary Points – Unit 6



## AppRouter

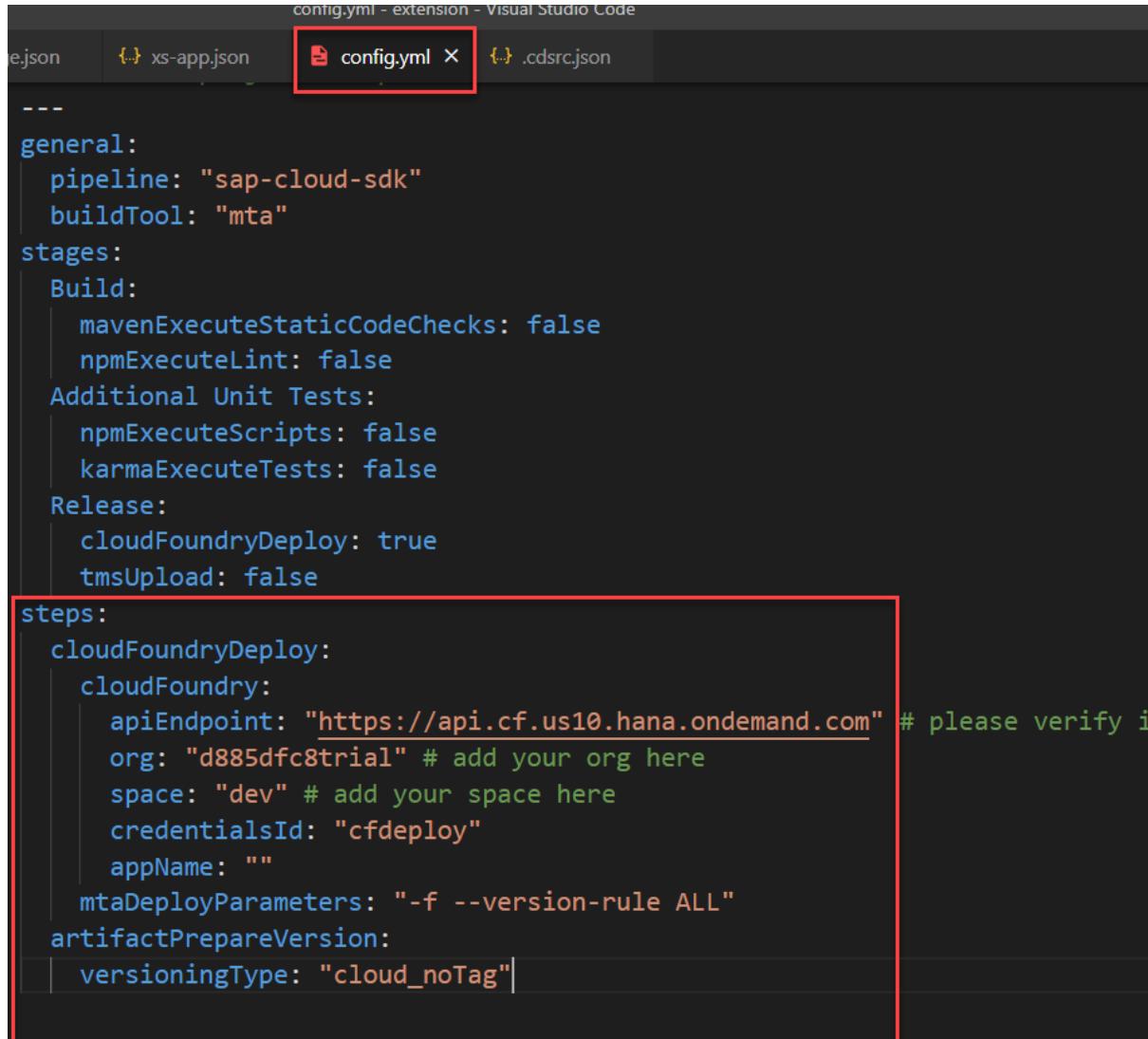
- Routes request from browser to CAP Service
- Routes request from browser to provider of UI sources
- Ensures authenticated and authorized users get token from XSUAA Service and forwards it to CAP Service

# Key Summary Points – Unit 7

[cds add pipeline](#)

## New Files

- [Jenkinsfile](#)
- [.pipeline/config.yml](#)



```
config.yml - extension - Visual Studio Code
---  
general:  
  pipeline: "sap-cloud-sdk"  
  buildTool: "mta"  
stages:  
  Build:  
    mavenExecuteStaticCodeChecks: false  
    npmExecuteLint: false  
  Additional Unit Tests:  
    npmExecuteScripts: false  
    karmaExecuteTests: false  
Release:  
  cloudFoundryDeploy: true  
  tmsUpload: false  
steps:  
  cloudFoundryDeploy:  
    cloudFoundry:  
      apiEndpoint: "https://api.cf.us10.hana.ondemand.com" # please verify i  
      org: "d885dfc8trial" # add your org here  
      space: "dev" # add your space here  
      credentialsId: "cfdeploy"  
      appName: ""  
      mtaDeployParameters: "-f --version-rule ALL"  
      artifactPrepareVersion:  
        versioningType: "cloud_noTag"
```

# Key Summary Points – Unit 3

***req.error, notify, info, warn*** (*code?, msg, target?, args?*)

Use these methods to collect messages or error and return them in the request response to the caller. The method variants reflect different severity levels, use them as follows:

## *Variants*

Method	Collected in	Typical UI	Severity
<code>req.notify</code>	<code>req.messages</code>	Toasters	1
<code>req.info</code>	<code>req.messages</code>	Dialog	2
<code>req.warn</code>	<code>req.messages</code>	Dialog	3
<code>req.error</code>	<code>req.error</code>	Dialog	4

**Note:** messages with severity < 4 are collected and accessible in property `req.messages`, while error messages are collected in property `req.errors`. The latter allows to easily check, whether errors occurred with:

```
if (req.errors) //> get out somehow...
```



The screenshot shows the SAP Studio interface with two main panes: the Explorer and the Schema Editor.

**Explorer:**

- RISK-MANAGEMENT
  - app
  - db
    - data
    - .gitkeep
  - schema.cds
  - node\_modules
  - srv
- .cdsrc.json
- .env-example
- .eslintrc
- .gitignore
- .npmrc
- LICENSE
- package-lock.json
- package.json
- README.md
- TROUBLESHOOTING.md

**Schema Editor (schema.cds):**

```
namespace riskmanagement;
using {
    managed,
    cuid,
    User,
    sap.common.CodeList
} from '@sap/cds/common';

entity Risks : cuid, managed {
    title           : String(100);
    owner           : String;
    prio            : Association to Priority;
    descr           : String;
    miti            : Association to Mitigations;
    impact          : Integer;
    // bp : Association to BusinessPartners;
    virtual criticality   : Integer;
    virtual PrioCriticality : Integer;
}

entity Mitigations : cuid, managed {
    descr      : String;
    owner      : String;
    timeline   : String;
    risks      : Association to many Risks
        |       |       |       on risks.miti = $self;
}

entity Priority : CodeList {
    key code : String enum {
        high   = 'H';
        medium = 'M';
        low    = 'L';
    };
}
```

# Key Summary Points – Unit 3

Q10. What is the main idea behind SAP Fiori elements?

**Choose the correct answer.**

- A Provide a framework and development tool kit for HTML 5.
- B Define a role-based user experience (UX).
- C Generate SAP Fiori apps at runtime from an existing OData service.
- D Provide a showcase for the core principles of modern user interfaces (UI).

**Q3.** Which of the following describes features of the SAP S/4HANA VDM?

---

- Documents the relationships between entities
- Enriches entities with business semantics
- Creates Smart UIs that are metadata driven
- D Provides a native UI to query the database tables

 Correct

You are correct! SAP S/4HANA VDM enriches entities with business semantics, creates Smart UIs that are metadata driven, documents the relationships between entities.

**Q2.** Which of the following options are the main use cases to be covered using side-by-side extensibility?



Creating a new UI for the existing application.



Changing the field's position in an existing screen.



Extending the existing application to include new functions.



Changing the colors of a standard UI.



Correct

That is correct! When using side-by-side extensibility, you must create a new UI for the existing application and extend the existing application including new functions.

# SAP Cloud SDK developer tools

A minimal set of prerequisites must be installed on our developer machine to build applications using the SAP Cloud SDK.

*Select each prerequisite to learn more.*



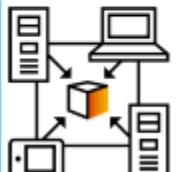
## Programming Language

Java SE Development Kit 8



## Command Line Development Tools

Maven, Git, Cloud Foundry CLI



## Integrated Development Environment

SAP Business Application Studio  
or Visual Studio Code  
or IntelliJ IDEA or Eclipse

## Programming Language

- The first thing we need is the Java Development Kit (JDK), for Java 8.
- The JDK is necessary for compiling Java applications and allows us to run Java applications locally.

# When to use SAP Cloud SDK ?

## Scenario 1

For the most part, when building applications using the SAP CAP Model, you will not even be aware that it is using the SAP Cloud SDK behind the covers. In fact, the recommendation in SAP CAP Model when connecting to remote services is not to directly leverage the SAP Cloud SDK. But there are times when you might want to directly leverage the SAP Cloud SDK from within the SAP CAP Model application. For example, if you have a requirement within your CAP application to retrieve the JWT token from the request object, then you can do so with the following code.

Code snippet

 Copy code

```
1
2 const { retrieveJwt } = require("@sap-cloud-sdk/core");
3 const jwt = retrieveJwt(req);
```

SAP CAP Model is fully compatible with SAP Cloud SDK. So even though, CAP Model hides the complexity to the developer, you could still directly leverage the SAP Cloud SDK to perform custom logic.

## Scenario 2

If you are planning to build your application using MongoDB, then you will not be able to use the CAP Model (since it only supports HANA for production). In this case, you will have to build your application without using the CAP Model. In such scenarios, more than likely you will need to use the SAP Cloud SDK for connecting to remote systems, consuming APIs in a type-safe manner etc. You can also use the SAP Cloud SDK to secure your application, provide multi-tenant support etc.

Of course, this scenario is not restricted to supporting MongoDB - but any scenario where you decide not to use the CAP Model for any reason. In all these scenarios, you will use the SAP Cloud SDK directly to build your extension applications.

## Scenario 3

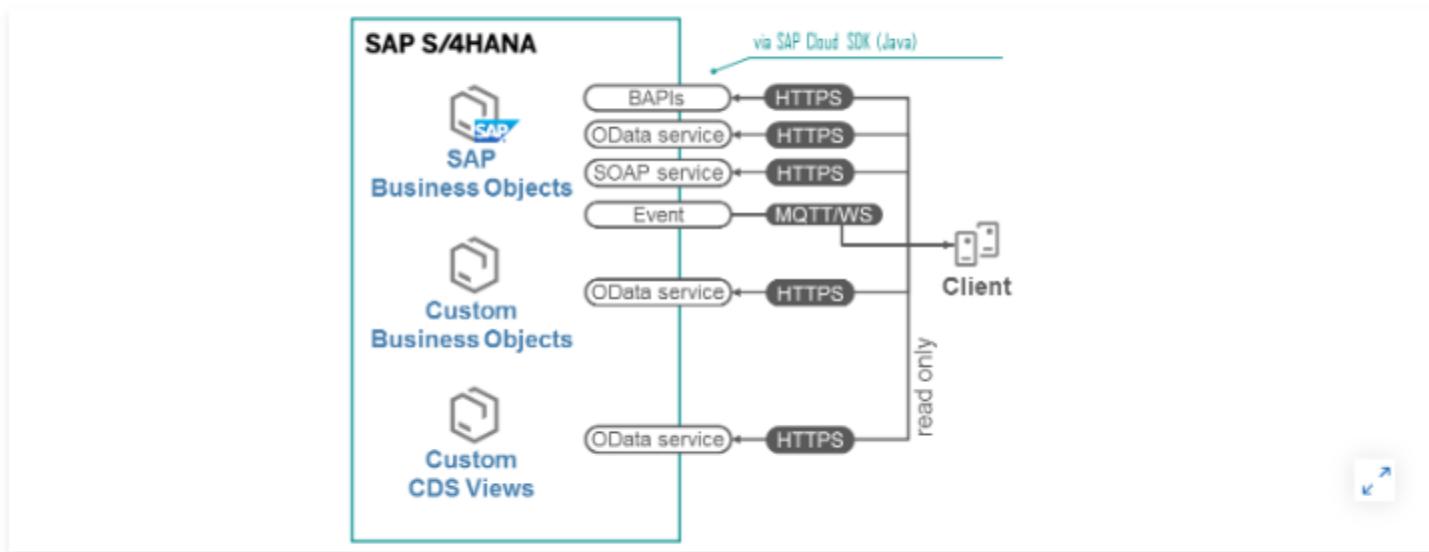
If you already have an existing Java application that was built without using the SAP CAP Model and you want to add more features to it. Here again, you can make use of the SAP Cloud SDK tool set.

## Scenario 4

If you need advanced features in your app (currently, not provided by CAP Model) - for example, Resilience. SAP Cloud SDK also allows you to generate the Virtual Data Model (VDM) of custom OData Services as well as standard SAP OData Services. This VDM can then be readily consumed by your application.

## Introducing SAP APIs

SAP provides a set of APIs that allow side-by-side extensions to interact with SAP S/4HANA Cloud. These APIs allow developers to read SAP data and perform actions that users can perform through the UI, such as getting a list of business partners or creating a new business partner address.



SAP APIs are lifecycle-stable, meaning they will continue to work the same way even when SAP S/4HANA Cloud is upgraded.

The APIs are also designed based on open protocols (OData, REST, and SOAP), to support the widest possible range of client libraries.

## The virtual data model

Due to the way the data is structured in SAP S/4HANA, it is difficult to query it manually. Consider a scenario where you need modeled data output related to the *Sales Order* transaction. Such cases would involve joins and queries to multiple database structures to get the desired results. Therefore, we need a tool that can ease the way we query the data and provide the desired results in a structured format. This is where the SAP S/4HANA virtual data model (VDM) comes into the picture.

SAP S/4HANA introduced a VDM that aims to remove this complexity and provide data in a semantically meaningful and easy to consume way. SAP S/4HANA shields the existing primary ERP structure/tables with an understandable, comprehensive logical data model.

Some of the features of the VDM include the following:

- User-known business terminology
- Documenting the relationships between entities
- Enriching entities with business semantics
- Metadata-driven creation of smart UIs

The VDM is built using the technology of SAP ABAP Core Data Services.

# SAP Cloud SDK – NestJs project

## Scaffold an Application

You can now create a new project using the command below:

```
nest new <project-name>
```

This will create an application that already contains all the files and configuration you need to use the SAP Cloud SDK for JavaScript. The CLI will ask you to select a package manager. Select "npm". The CLI will then install all the necessary dependencies for the project, so this might take a minute. If everything worked correctly, you should see an output like this:

```
🚀 Successfully created project <project-name>
👉 Get started with the following commands:
$ cd <project-name>
$ npm run start
```

## Run the Project

To run the application locally, execute the following command:

```
npm run start:dev
```

This will start a local server in watch mode so that subsequent changes will automatically trigger a restart of the server. Go to <http://localhost:3000> and you should get a "Hello World!" in response. Before continuing with the next part of the tutorial, open `src/main.ts` and switch the port from 3000 to 8080. This is required as the mock server (covered [later](#)) runs on port `3000`. The corresponding line should then look like this:

```
await app.listen(process.env.PORT || 8080);
```