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 replicasets

Kubernetes API via kubectl

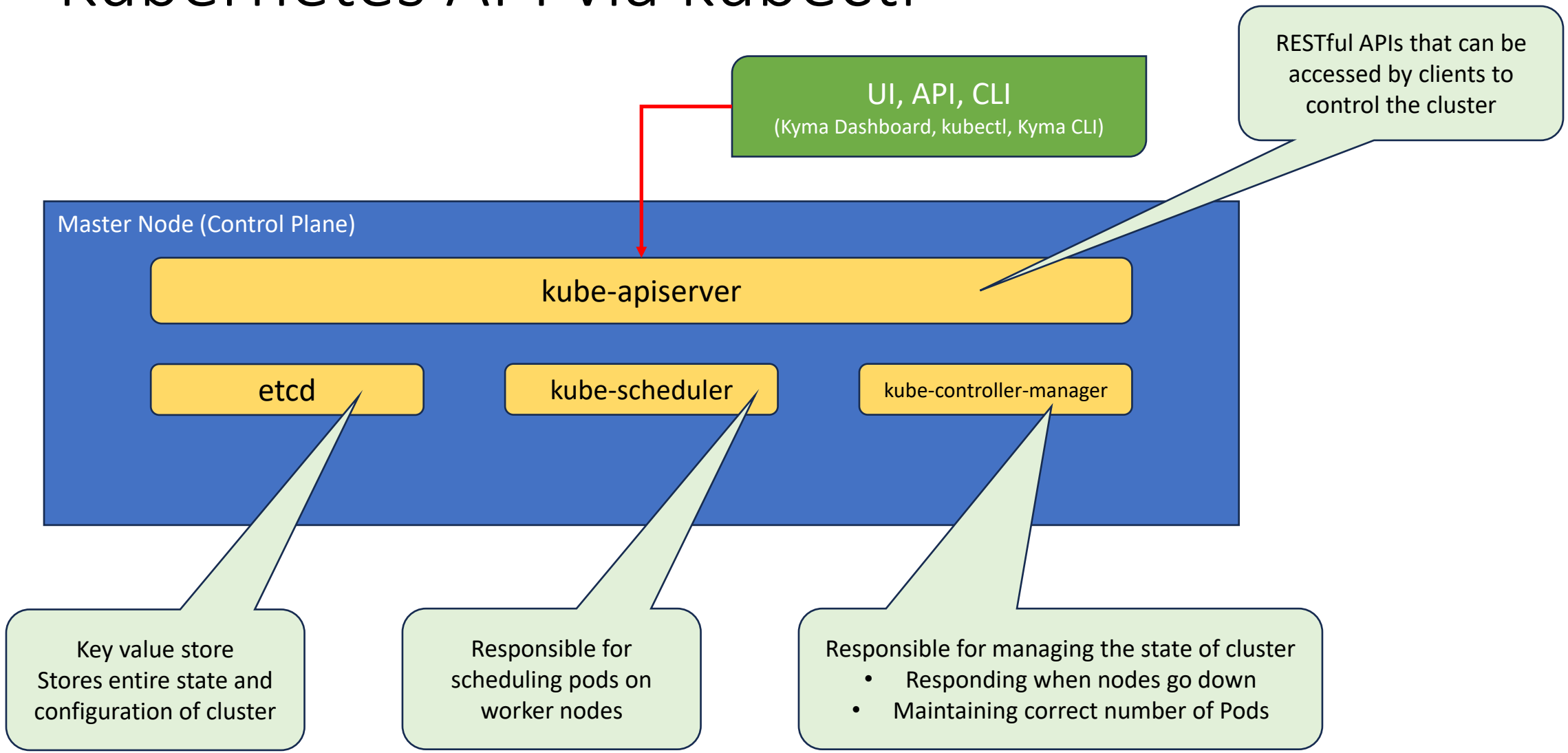
Q1. Which command can you use to create a pod from a YAML file?

- ✓ kubectl create -f my-pod.yaml
- B kubectl create my-pod.yaml
- C kubectl upload my-pod.yaml

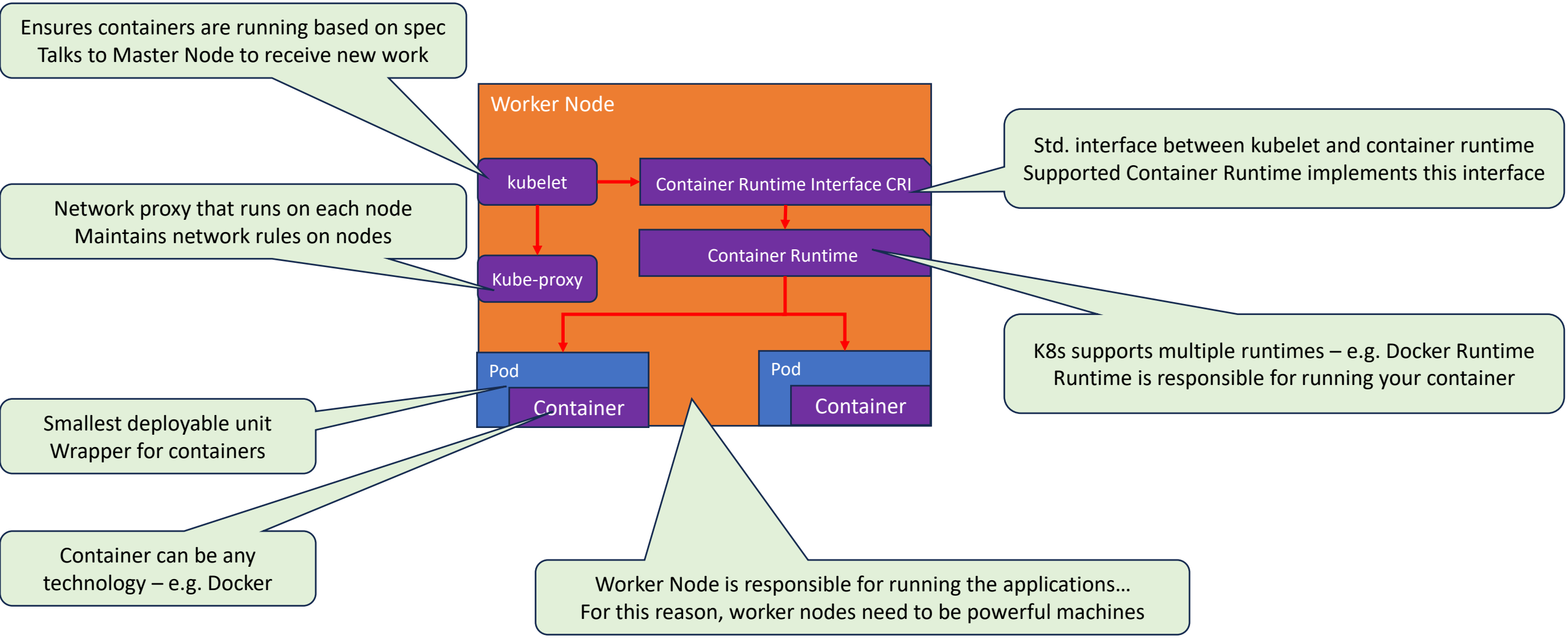
✓ Correct

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

Kubernetes API via kubectl



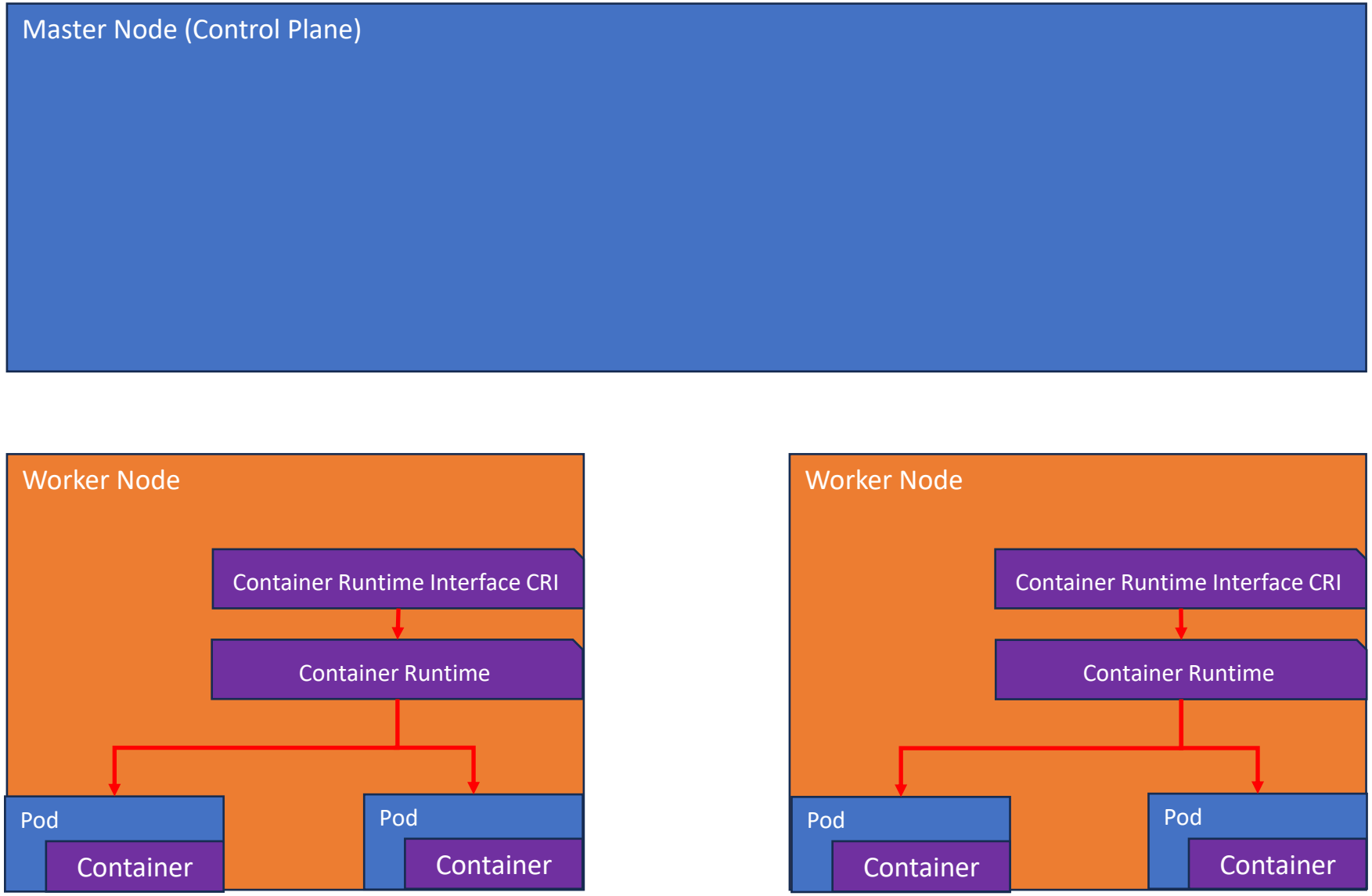
Kubernetes API via kubectl



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

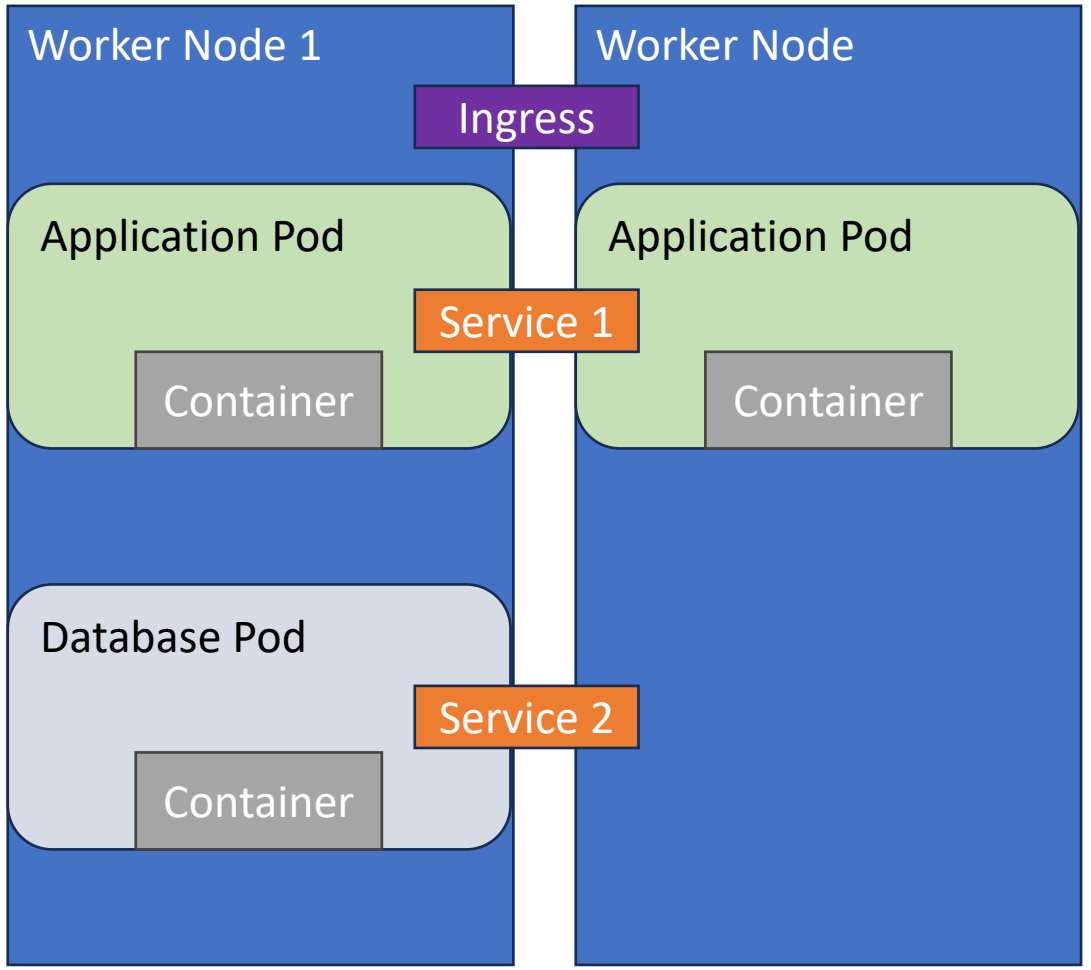
ReplicaSets define how many Pods we need in the cluster

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

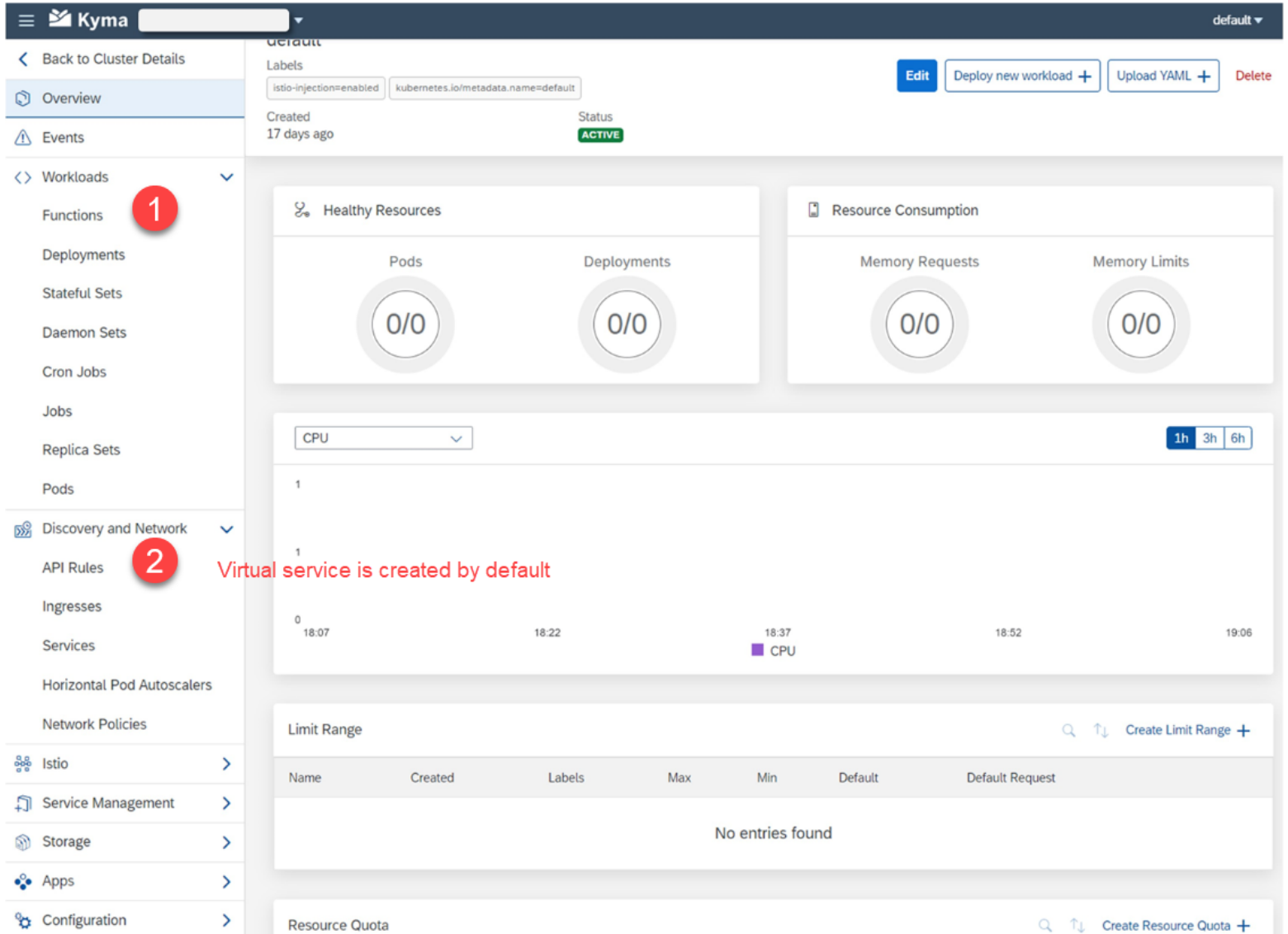
C Services are a way to group Deployments together

D Services are a way to group ReplicaSets together

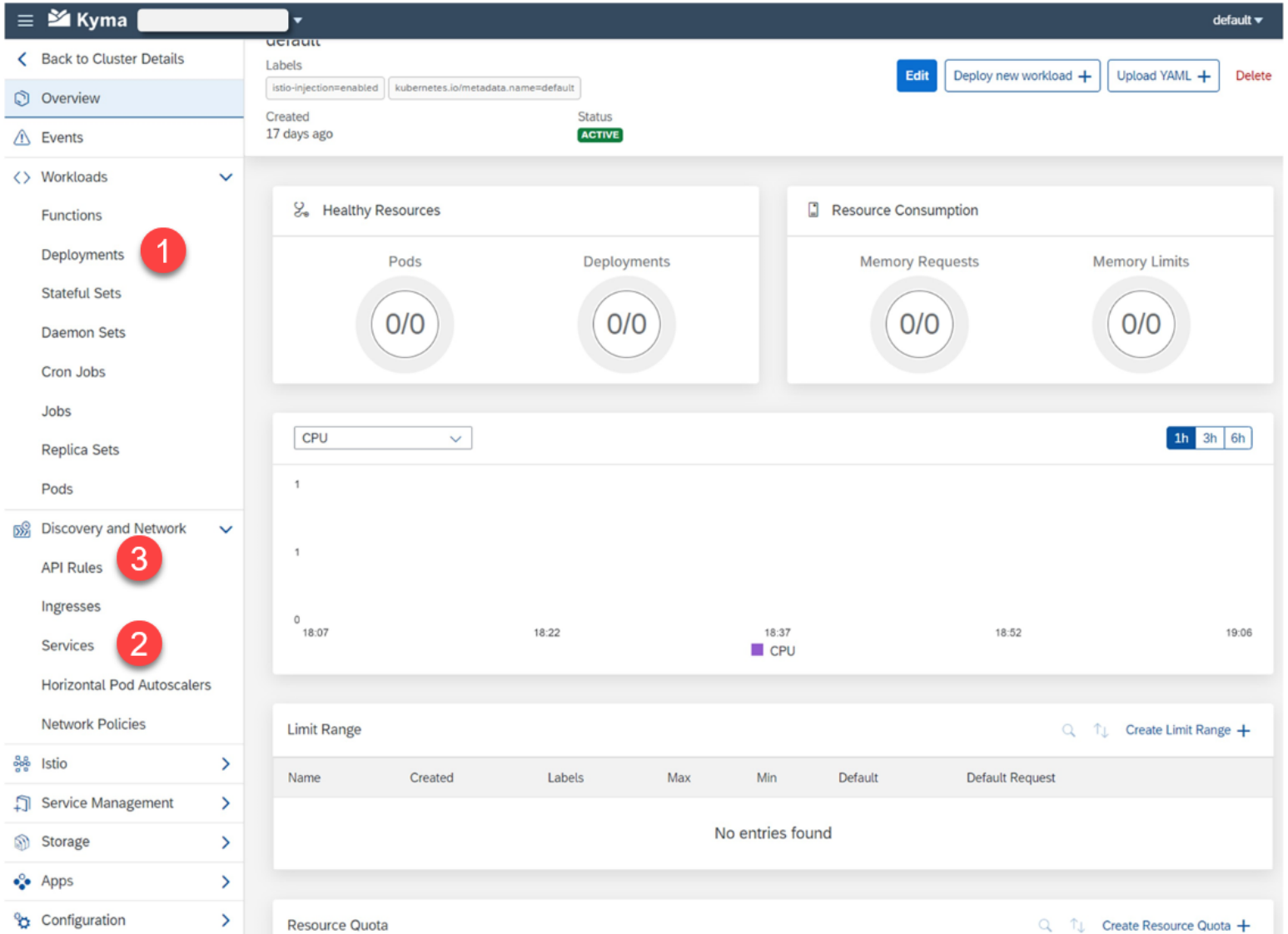
✓ Correct

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

Serverless Functions



Deployments



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

✓ 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.

✓ Istio's components are installed in the istio-system namespace.

✓ The APIRule custom resource is translated into Istio components (VirtualService).

D A Kubernetes Ingress translates into Istio components (VirtualService).

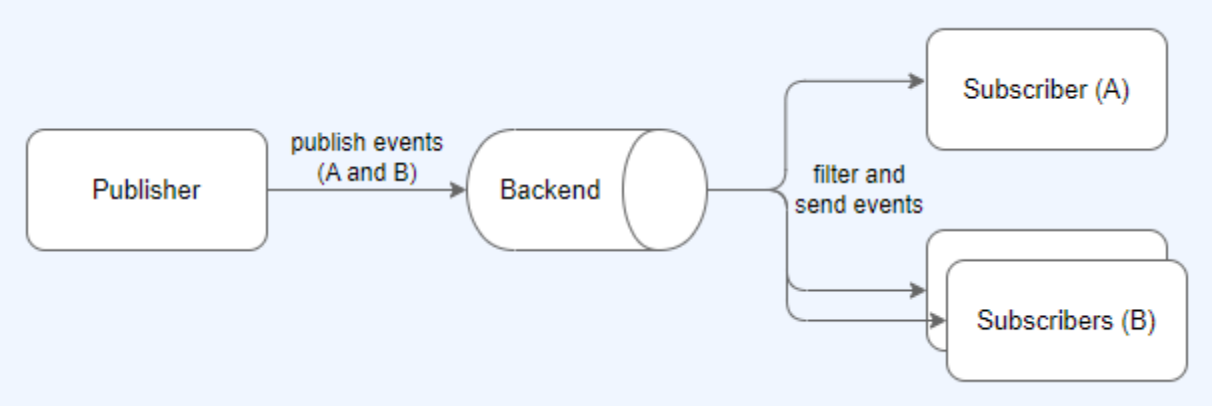
✓ Correct

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

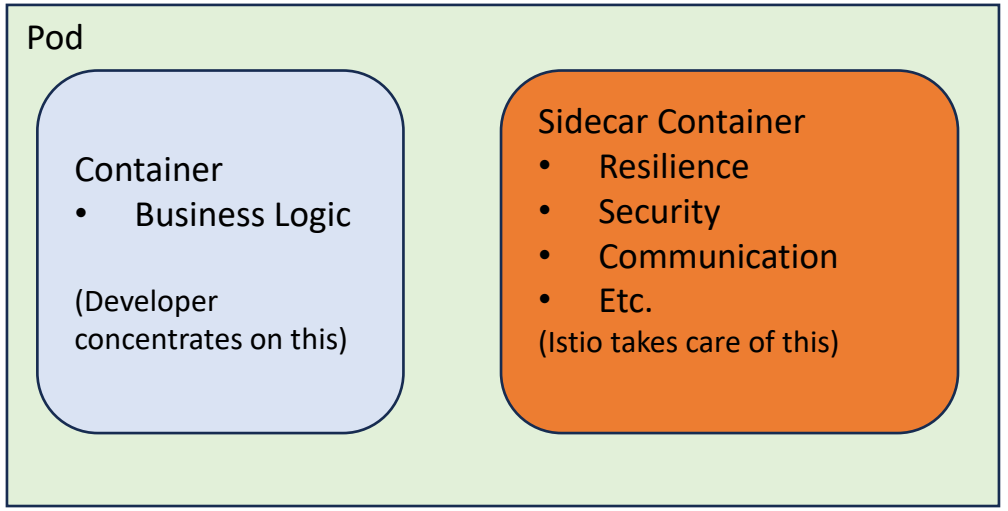


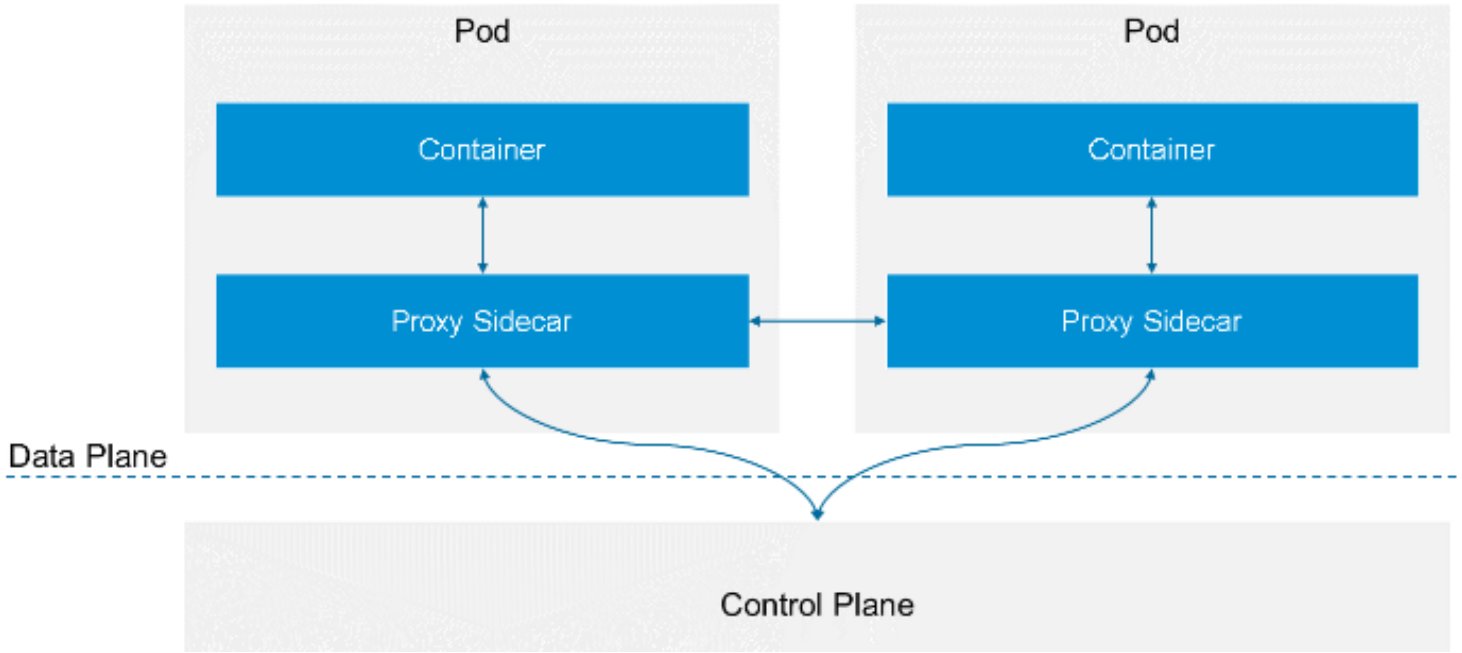
Eventing in Kyma – how it works ?

- 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

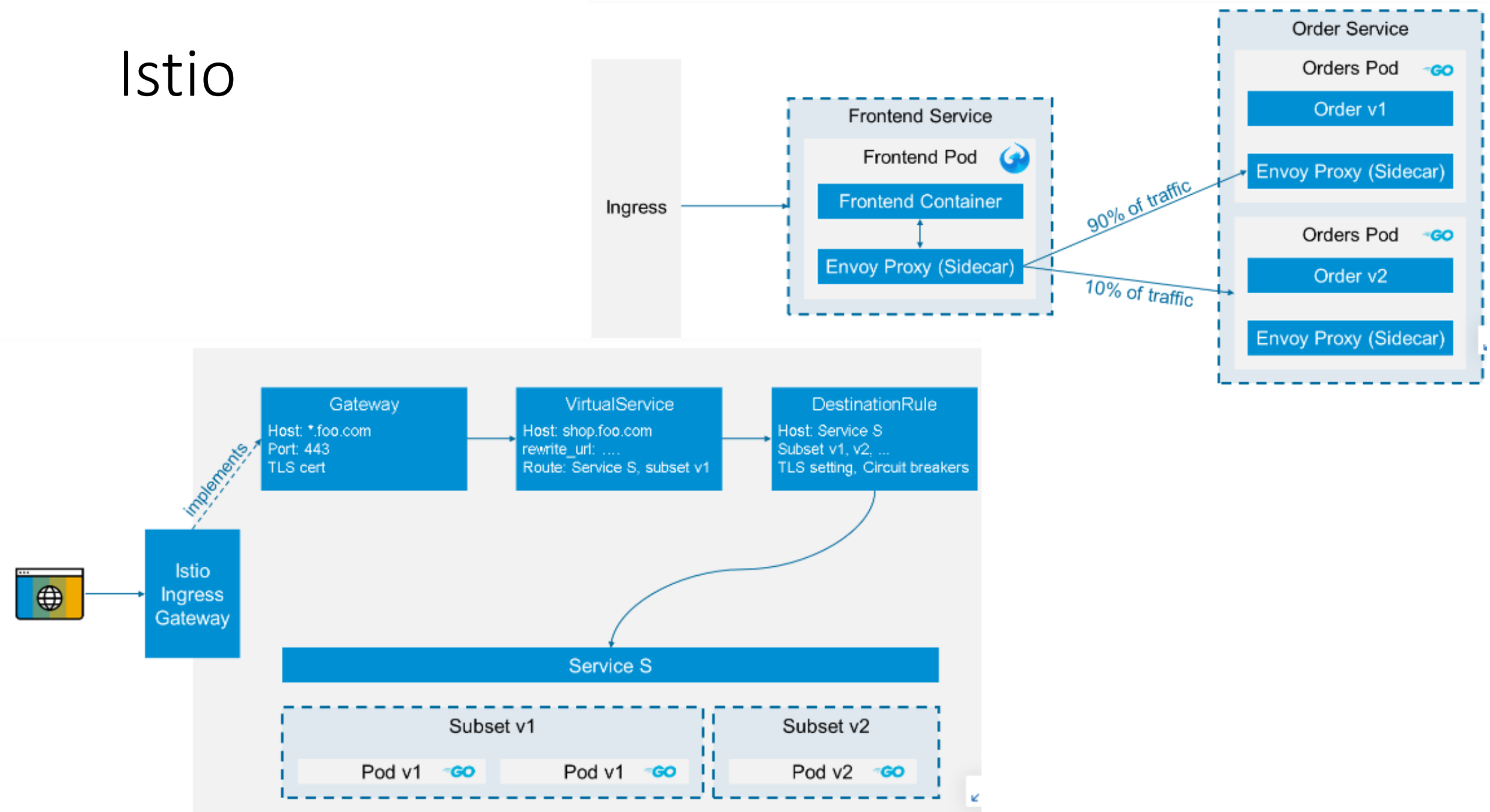
Discovering Service Mesh

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

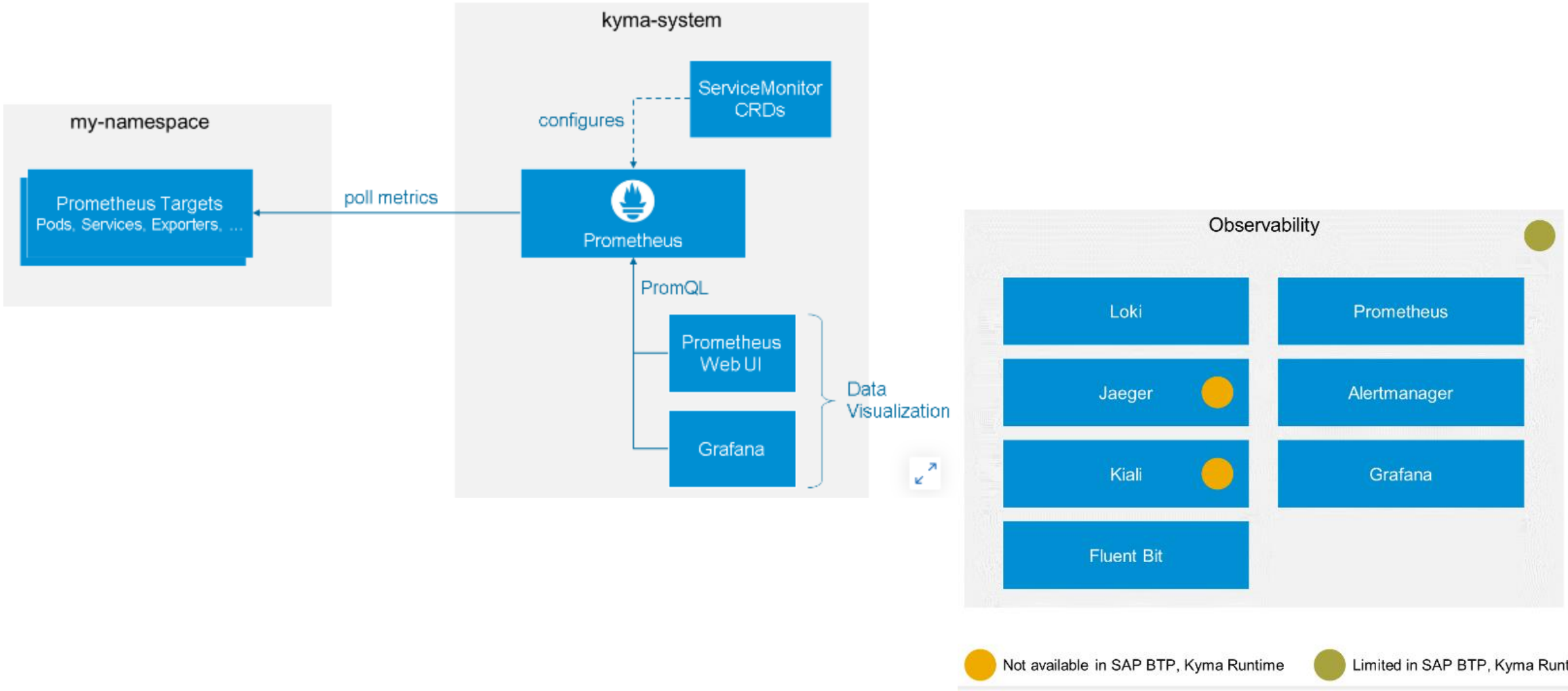




Istio


Ingress
Frontend Service
Frontend Pod
Frontend Container
Envoy Proxy (Sidecar)
90% of traffic
10% of traffic
Order Service
Orders Pod
Order v1
Envoy Proxy (Sidecar)
Orders Pod
Order v2
Envoy Proxy (Sidecar)
Gateway
Host: *.foo.com
Port: 443
TLS cert
VirtualService
Host: shop.foo.com
rewrite_url: ....
Route: Service S, subset v1
DestinationRule
Host: Service S
Subset v1, v2, ...
TLS setting, Circuit breakers
implements
Istio Ingress Gateway
Service S
Subset v1
Pod v1
Pod v1
Subset v2
Pod v2

Observability and Monitoring



StatefulSet

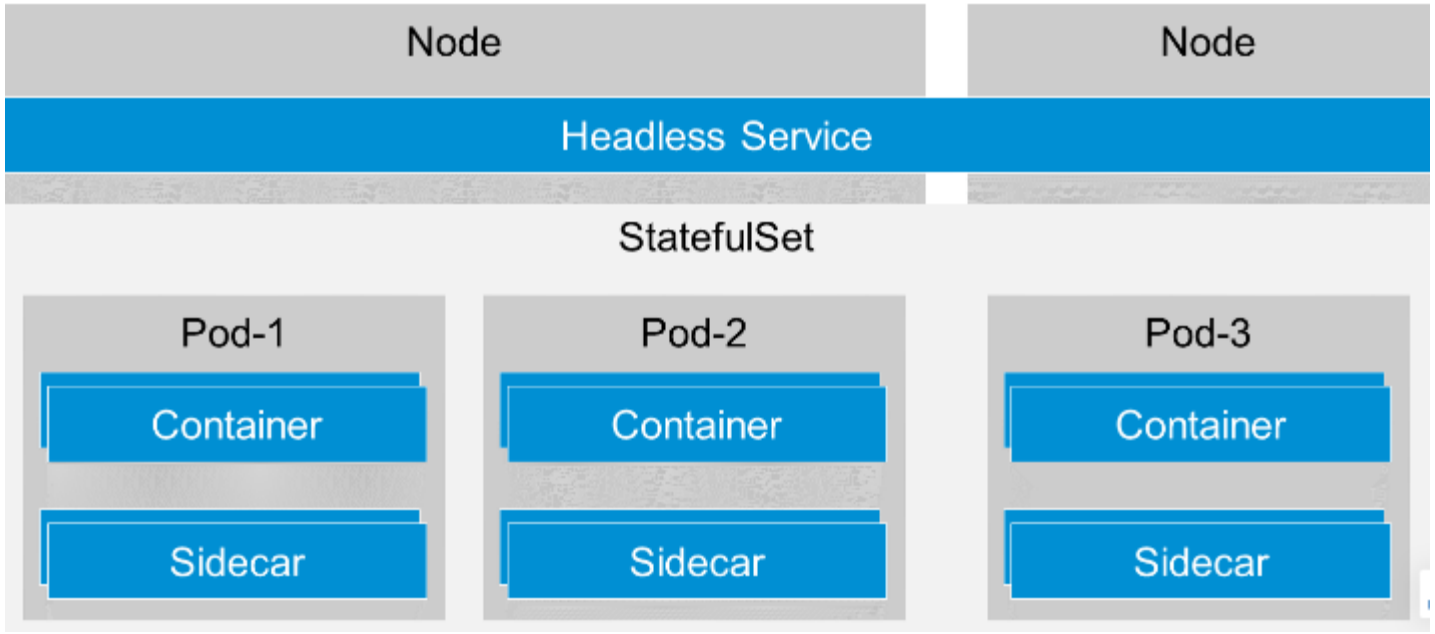
In most cases, pods are stateless
They die and are replaced by any other pod with some random name

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

✓ StatefulSet

D Pod

E ReplicaSet

✓ 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

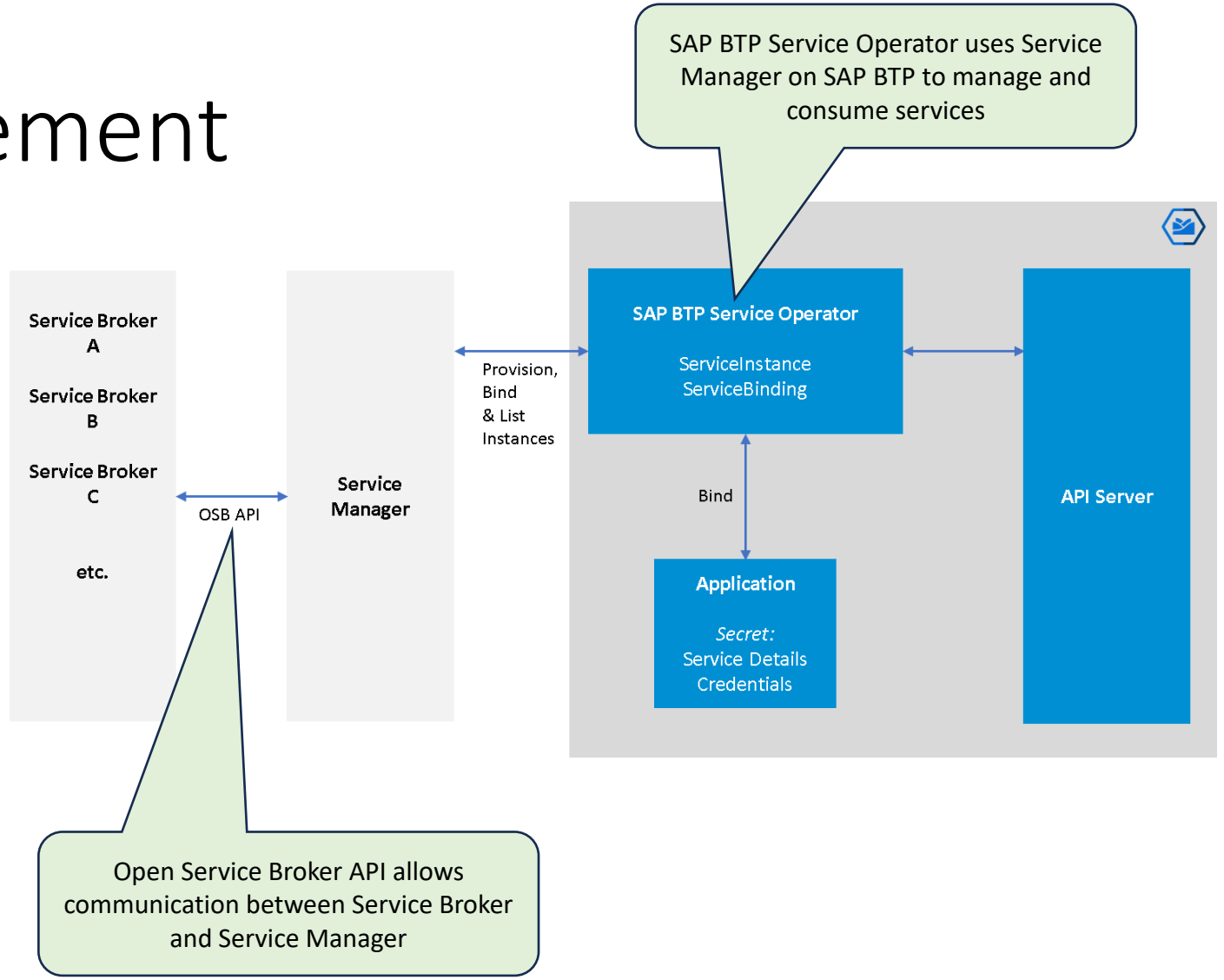
B Queue Broker API

C Messaging

✓ 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
6   data:
7       username: "YWRtaW4=" # This is the base64 encoded string "admin"
8       password: "dG9wc2VjcmV0" # This is the base64 encoded string "topsecret"
```

different types available...

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`:

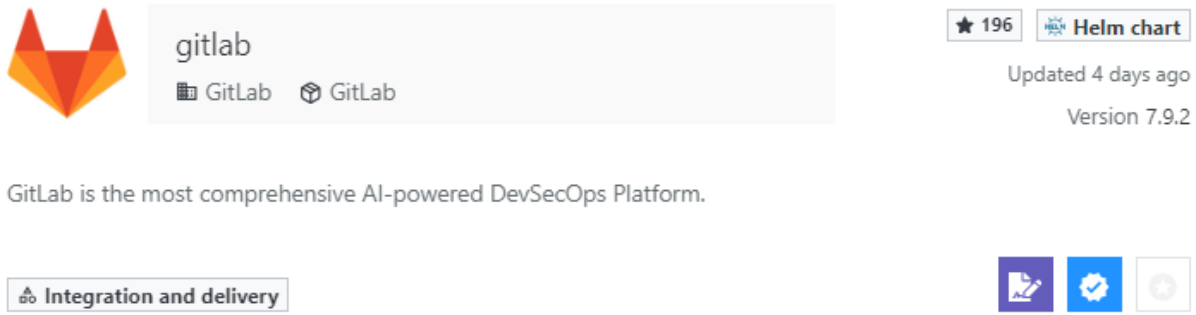
Code snippet

Copy code

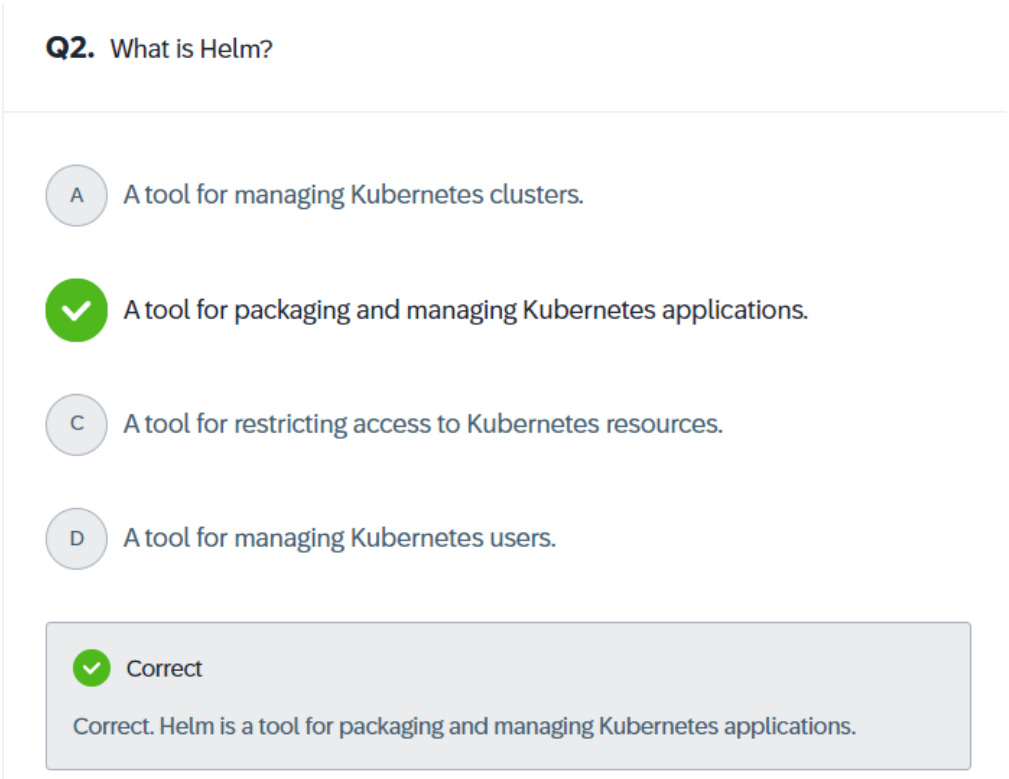
```
...
spec:
...
  template:
  ...
    spec:
      imagePullSecrets:
        - name: myregistrykey
      containers:
        - name: hello-kyma
          image: ghcr.io/sap-samples/kyma-runtime-learning-journey/hello-kyma:1.0.0
          ports:
            - containerPort: 8080
```

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*



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

```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 benefits 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 a 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

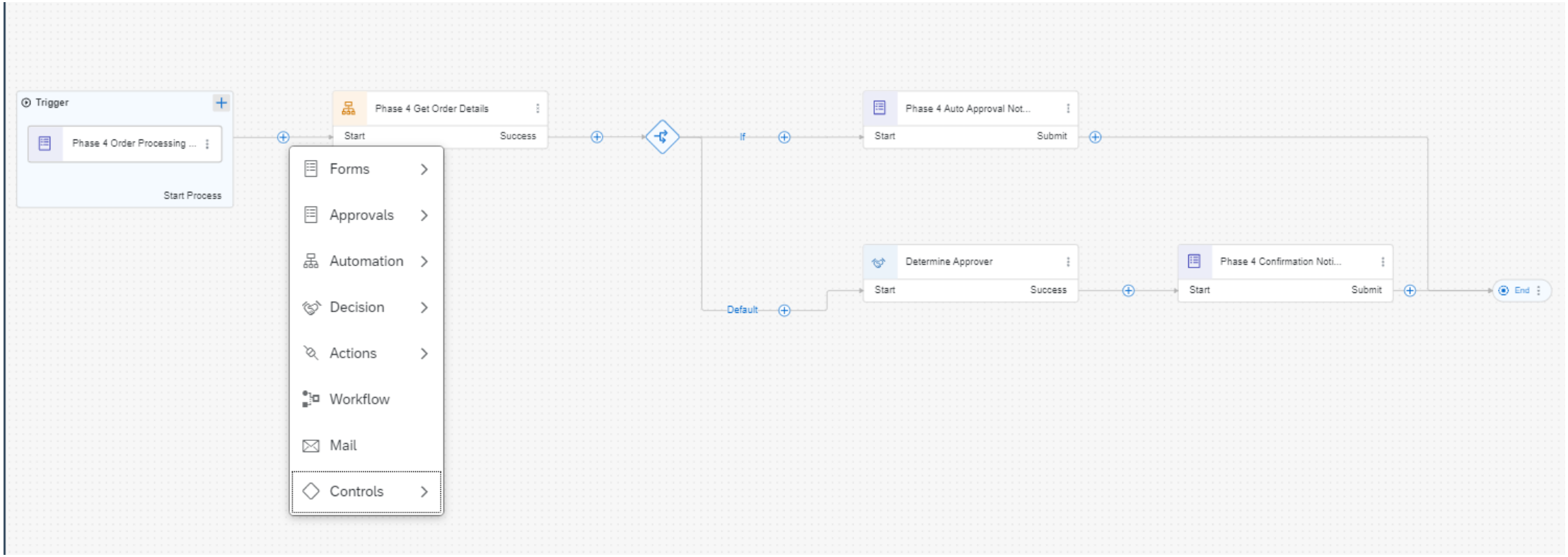
Q1. Which of the following artifacts are usually part of an added Live Process from the SAP Build Process Automation Store?

- ✓ Decision
- ✓ Visibility Scenario
- C Preconfigured APIs
- D 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

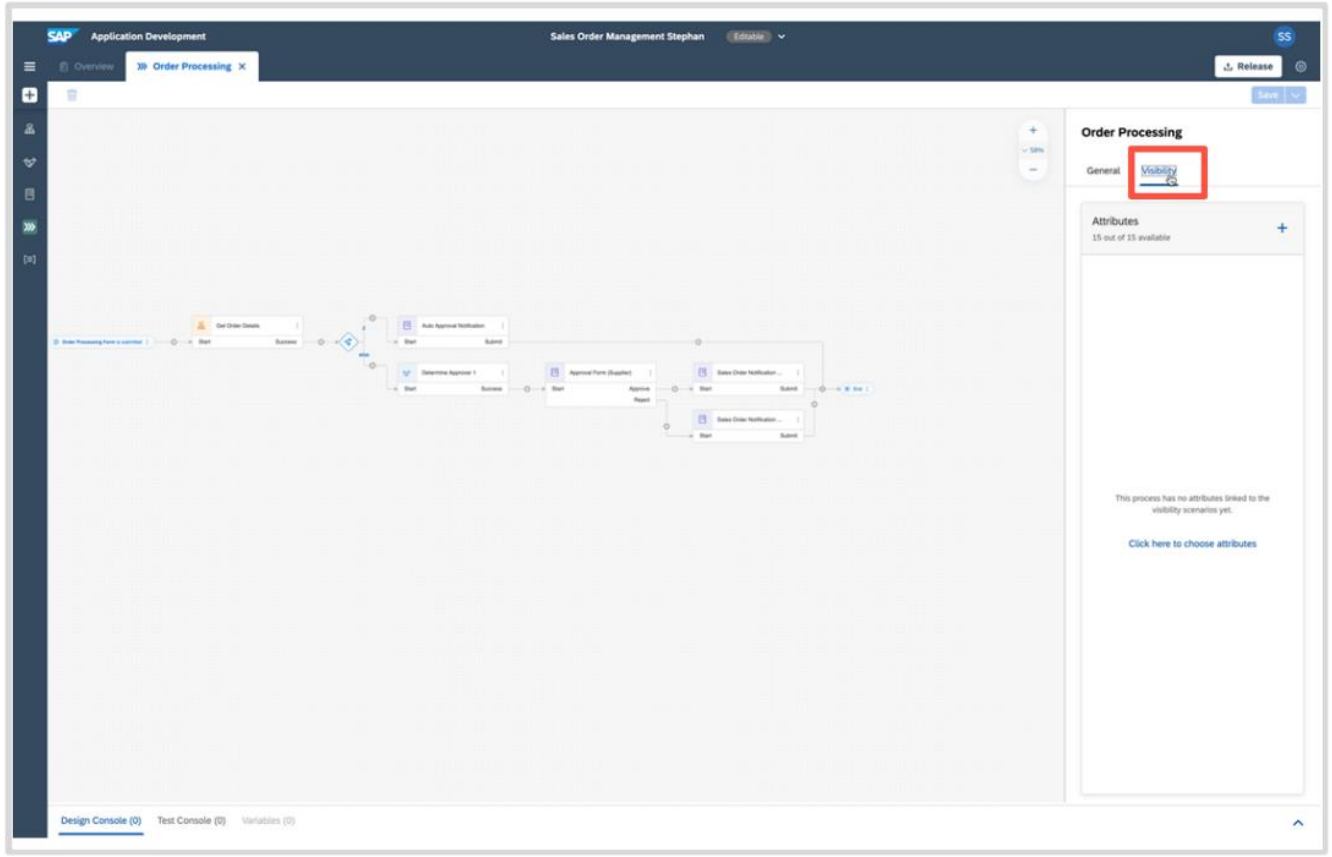
Q6. Which configurations are necessary for Process Conditions?

- ✓ Add a condition to the process
- B Map the condition input with process content
- ✓ Configure If and else conditions
- ✓ Define the process flow for different conditional paths
- E Add environment variables

✓ 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**.

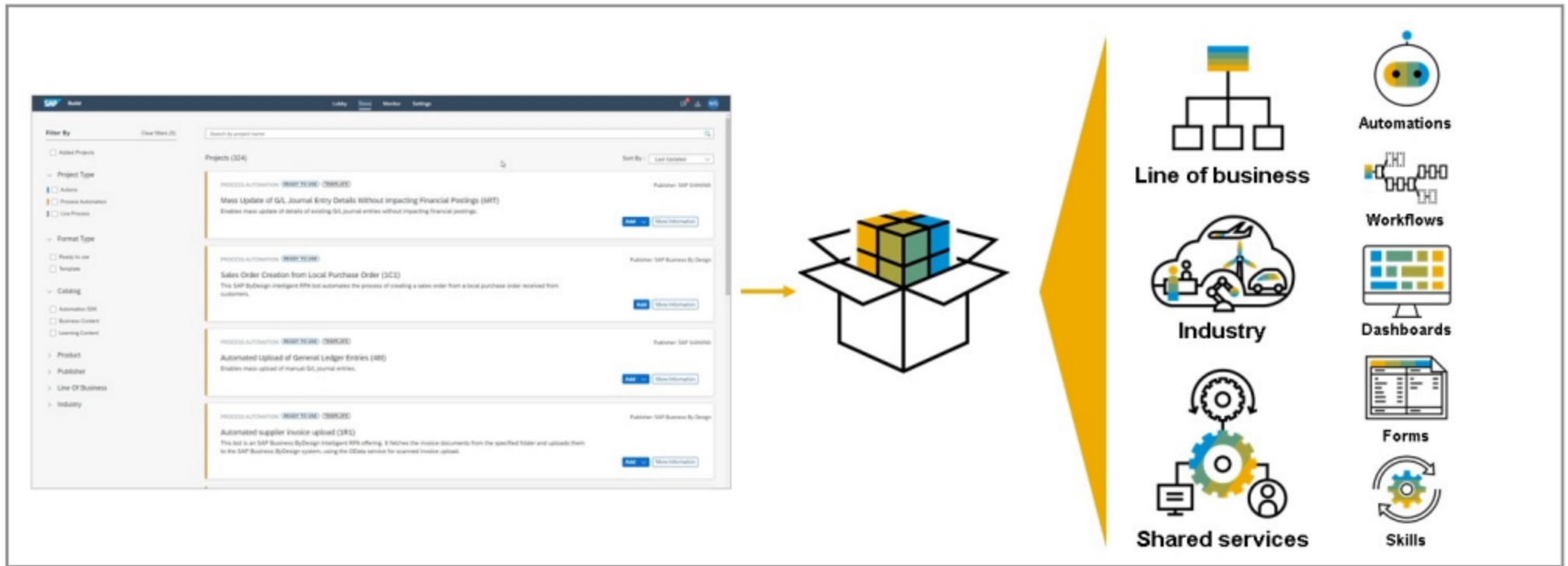
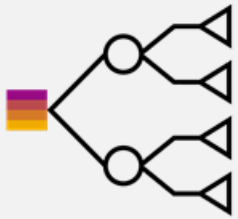


Figure 2: Import of Prebuilt Processes and Automations

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.

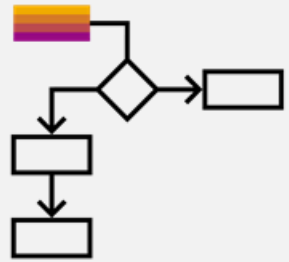


Decision

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.

Rules

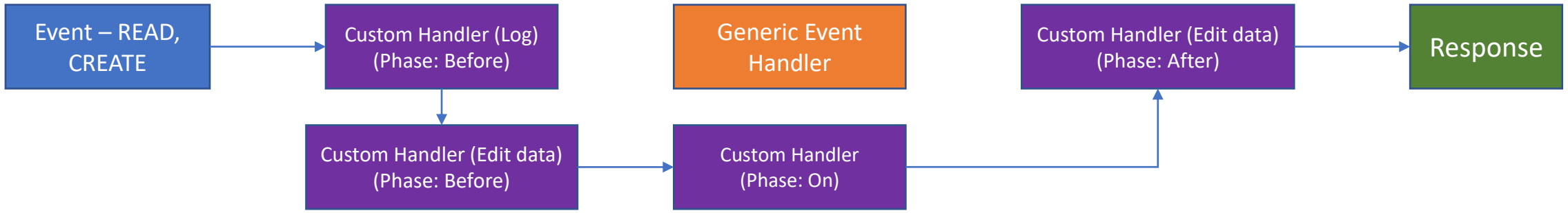
Decision Table Rule

Text Rule

Decision Table / Text Rules

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

- The 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';
2
3  @path : 'service/risk'
4  service RiskService {
5      entity Risks       as projection on rm.Risks;
6      annotate Risks with @odata.draft.enabled;
7
8      entity Mitigations as projection on rm.Mitigations;
9      annotate Mitigations with @odata.draft.enabled;
10 }
11
```

Point to where the schema.cds file is located... Alias of rm is used

Override the URL path... Line below creates an OData service named RiskService

This line adds the entity to the RiskService... Includes all the fields

This line enables draft functionality

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)

- [x] YAML uses whitespace indentation for structuring purposes.
- [ ] YAML uses tab indentation for structuring purposes.
- [ ] YAML uses hyphens: - for comments.
- [x] 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.

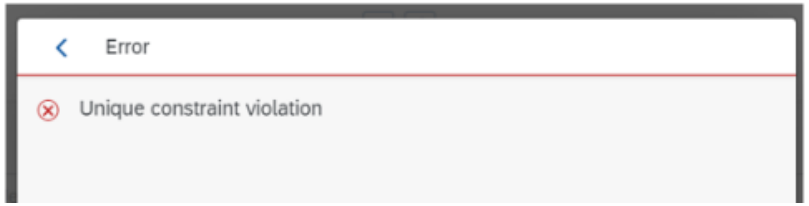
Never try to catch and handle unexpected errors, rejections of promises, and so on. If it is unexpected, you cannot handle it correctly. If you could, it would be expected (and should already be handled). Even if your apps should be stateless, you can never be 100% sure that a shared resource was not 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

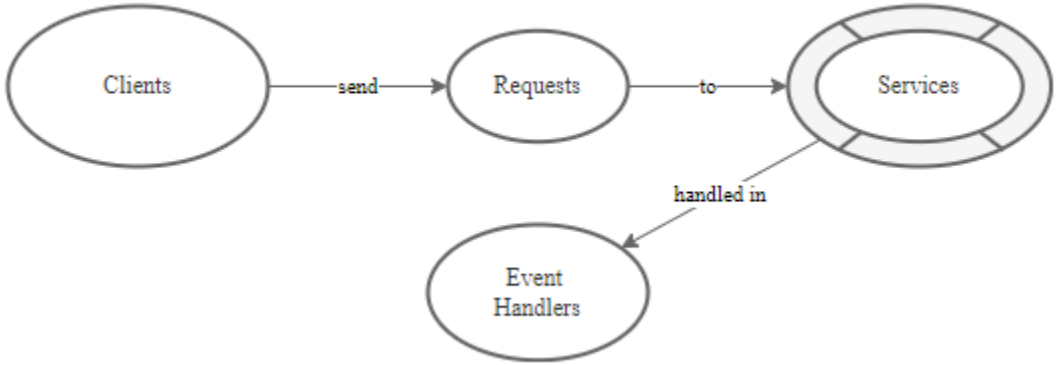
✓ Register an error handler

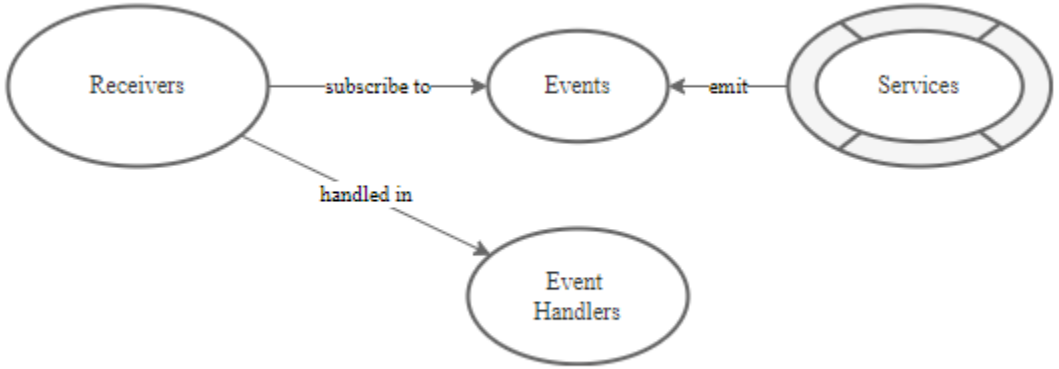
D Register an event handler

✓ Correct

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

Key Summary Points – Unit 3

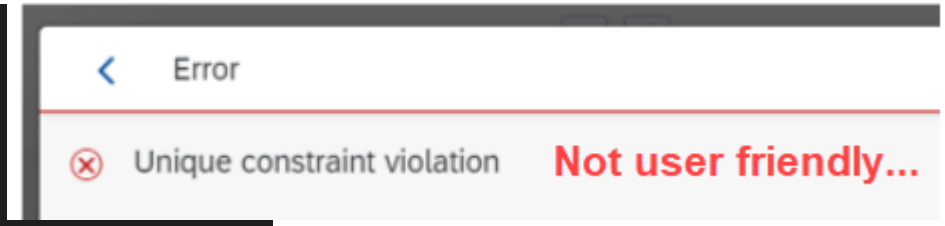


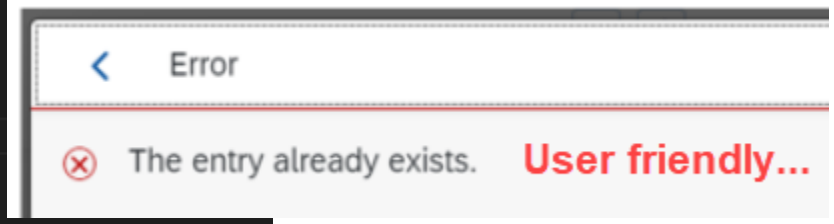


Error Handling

```
/**
 * 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 occured. 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

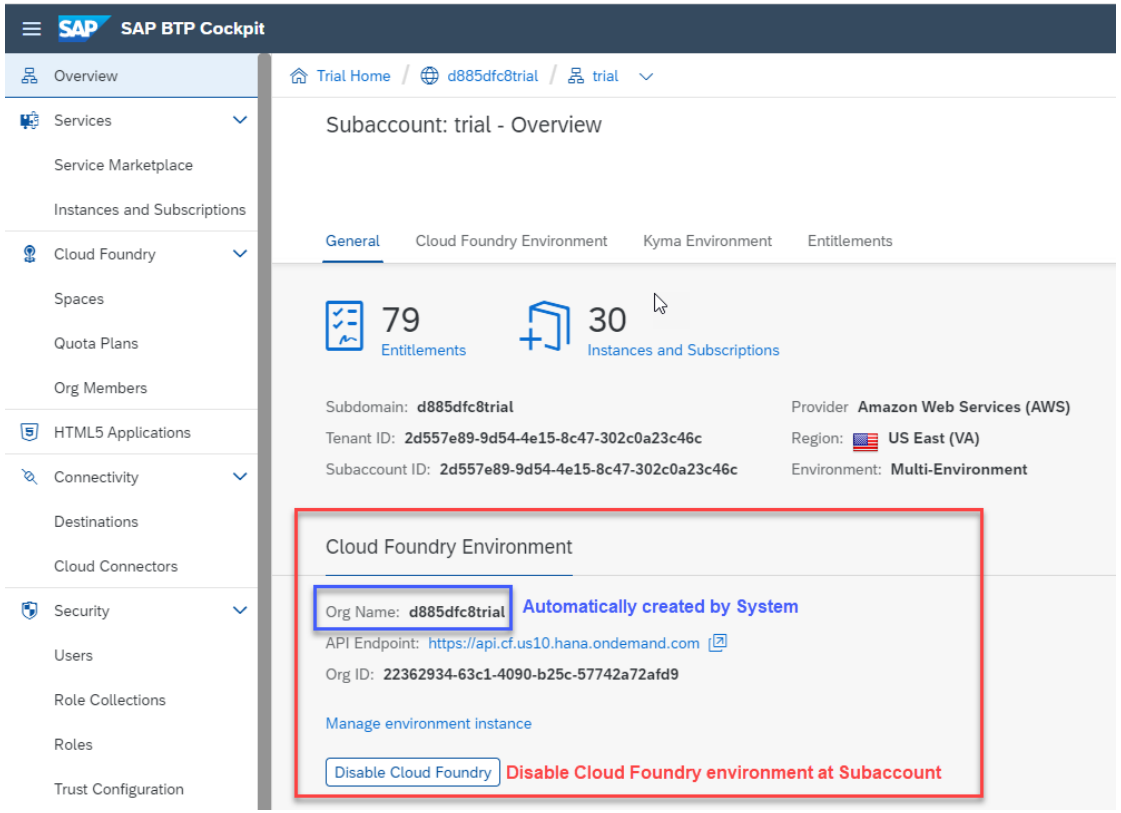
I want to…	Services required
Connect to publicly available Services	Destination Service
Connect to On-Premise Services	Destination Service, Connectivity Service

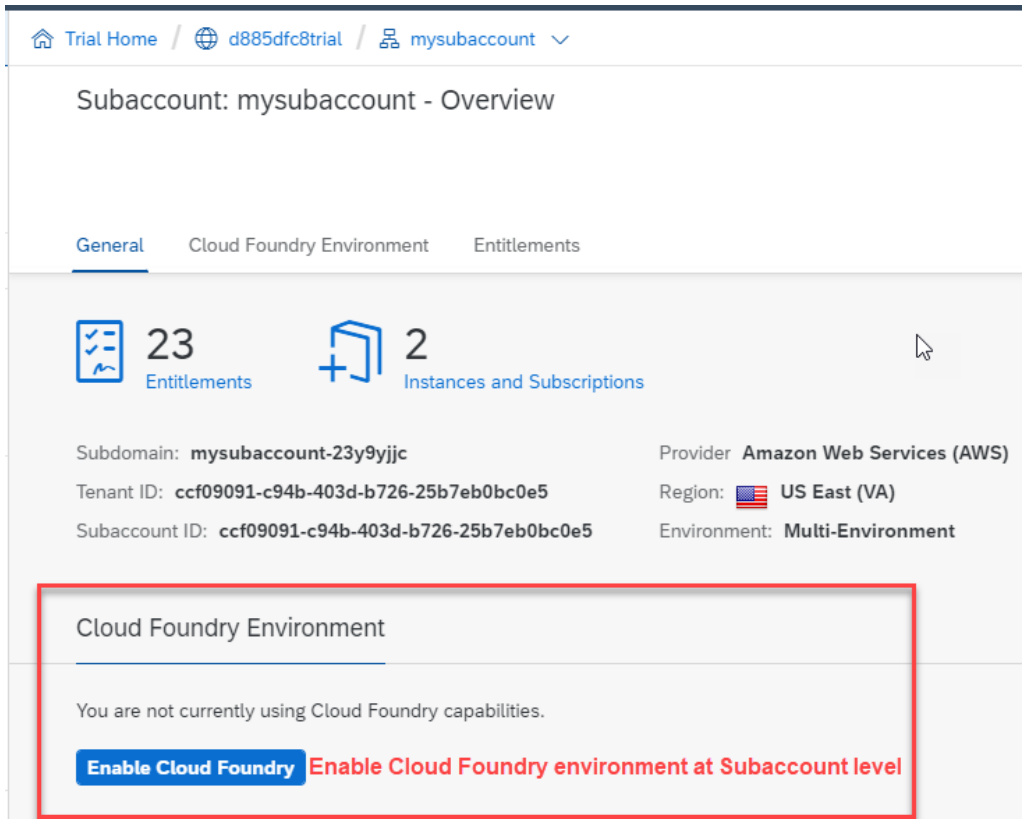
Key Summary Points – Unit 4

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.

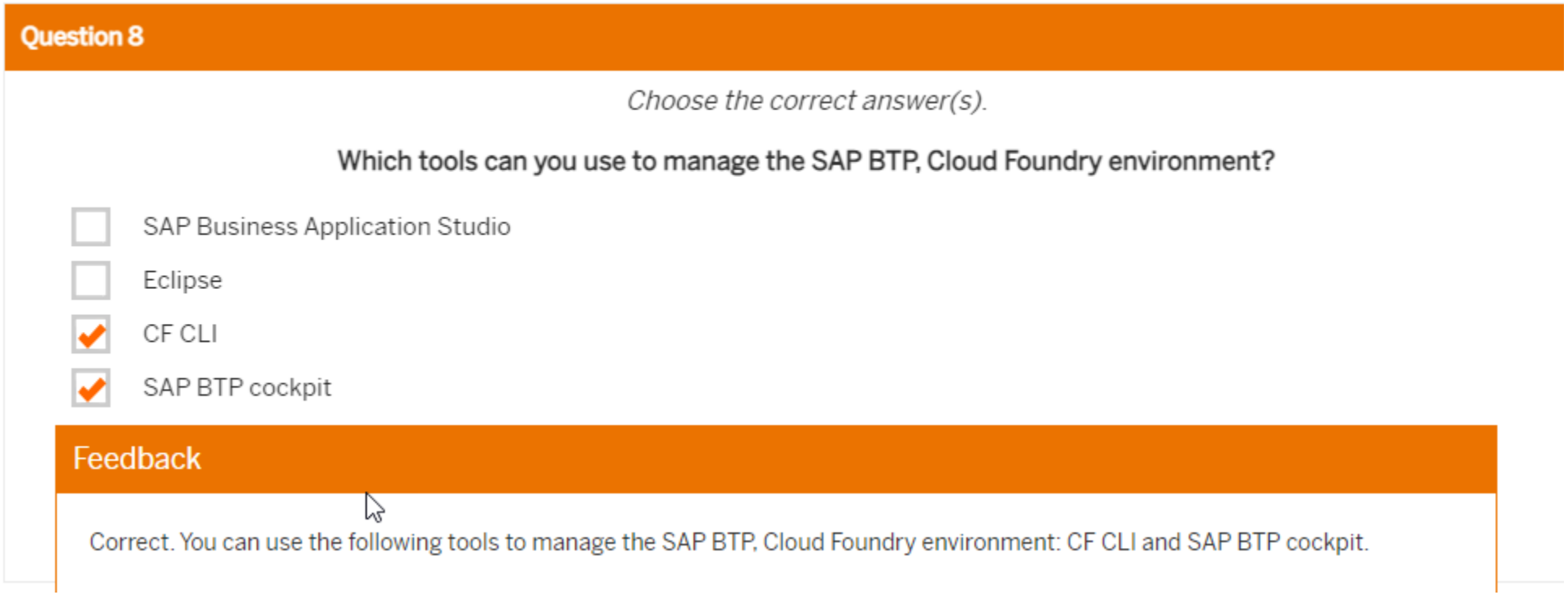
Key Summary Points – Unit 5





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



Key Summary Points – Unit 6

```
service RiskService {
    entity Risks @(restrict : [
        {
            grant : ['READ'],
            to    : ['RiskViewer']
        },
        {
            grant : ['*'],
            to    : ['RiskManager']
        }
    ])                      as projection on rm.Risks;

    annotate Risks with @odata.draft.enabled;

    entity Mitigations @(restrict : [
        {
            grant : ['READ'],
            to    : ['RiskViewer']
        },
        {
            grant : ['*'],
            to    : ['RiskManager']
        }
    ])                      as projection on rm.Mitigations;

    annotate Mitigations with @odata.draft.enabled;
```

restrict - keyword for Risks entity

RiskViewer Role - Can only READ

RiskManager Role - Can do everything

restrict - keyword for Mitigations entity

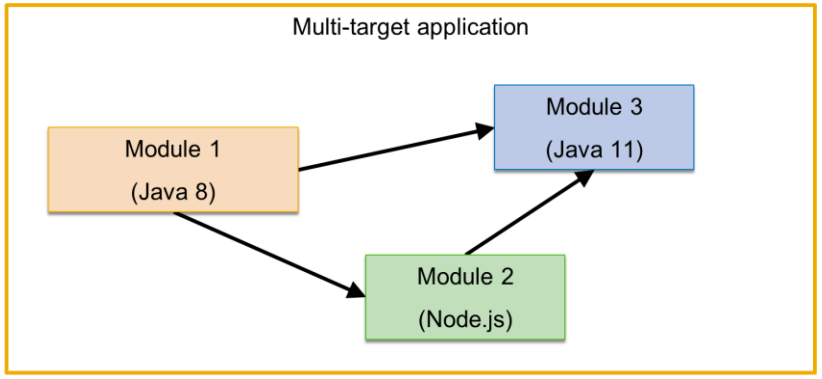
RiskViewer Role - Can only READ

RiskManager Role - Can do everything

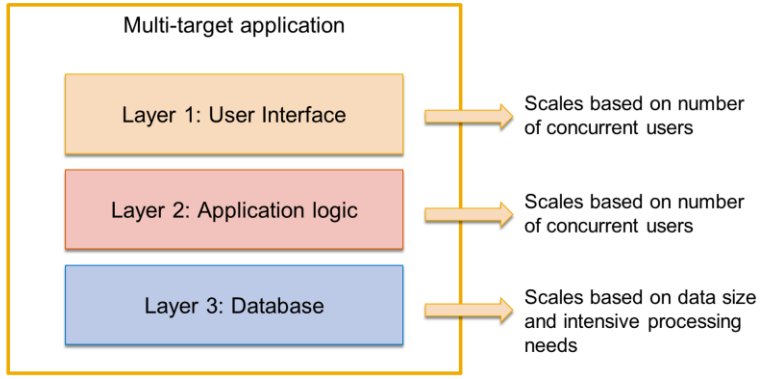
srv/risk-service.cds

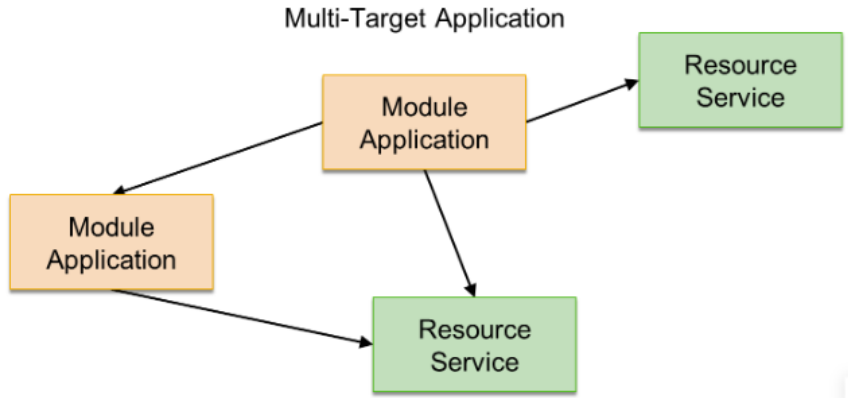
Multi Target Applications

Use MTA to integrate different programming languages or runtime environments



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





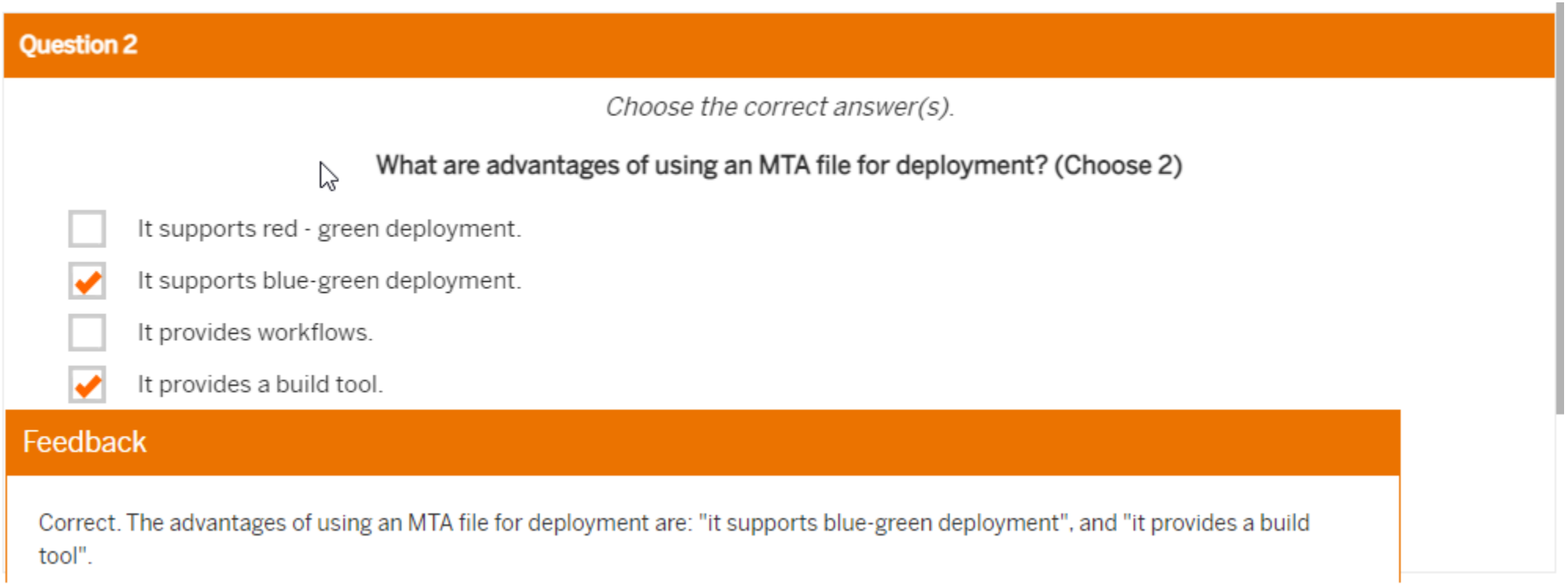
Q1. Which tool do you use to build an MTA?

- mbt
- B multiapps
- C Maven
- D 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.
- [x] It supports blue-green deployment.
- [ ] It provides workflows.
- [x] It provides a build tool.

Feedback

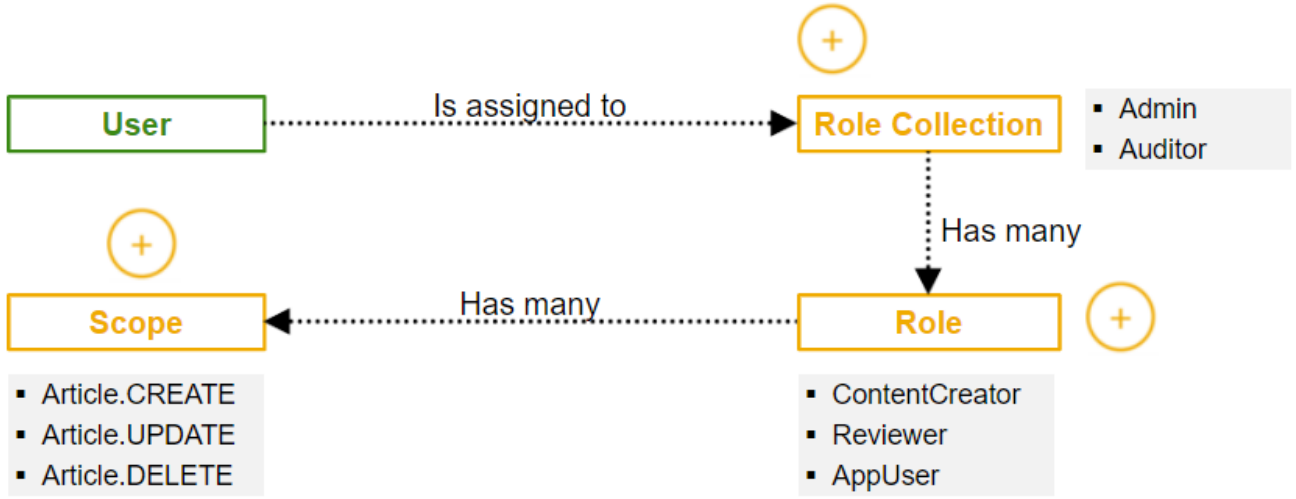
Correct. The advantages of using an MTA file for deployment are: "it supports blue-green deployment", and "it provides a build tool".

mbt build –t ./
mbt build

Key Summary Points – Unit 6

```
"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

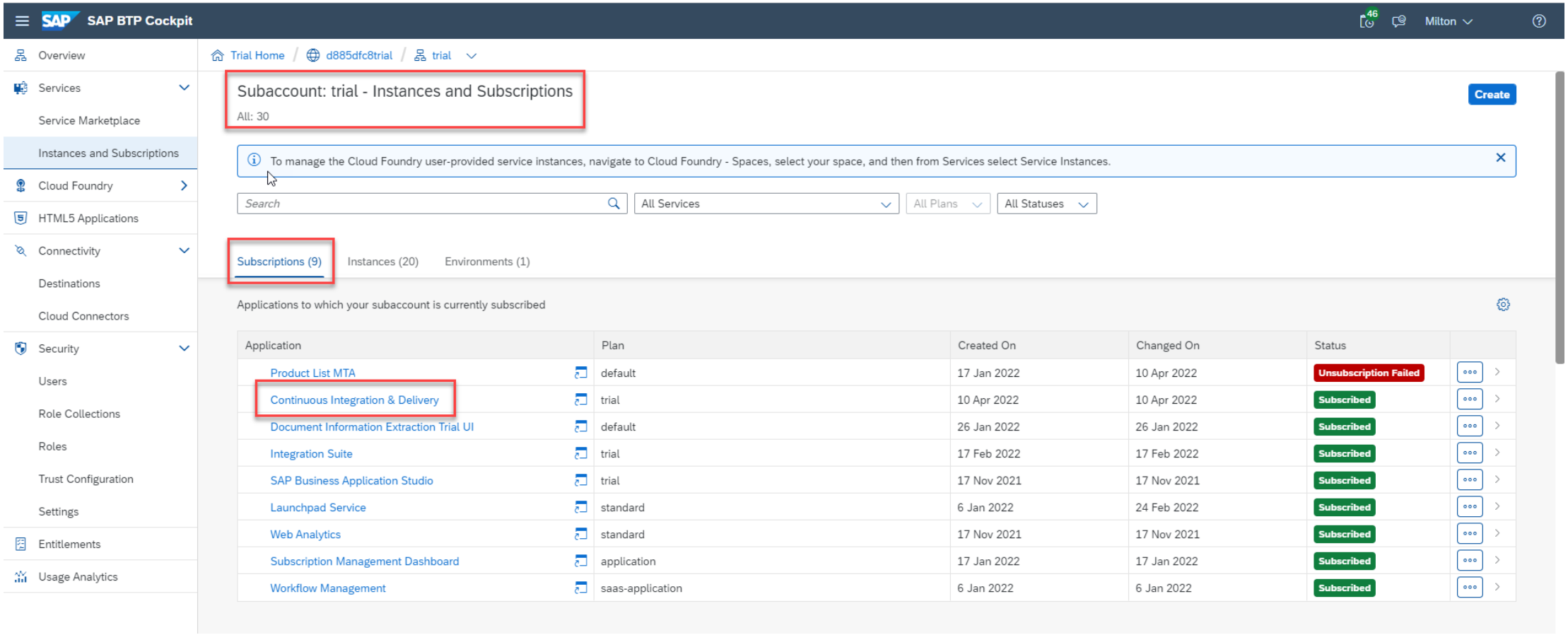


- In SAP BTP, CF environment, a single authorization is called Scope
- Scopes cannot be assigned to users directly – They are packaged into Roles
- Scopes are prefixed with xsappname to make them uniquely identifiable

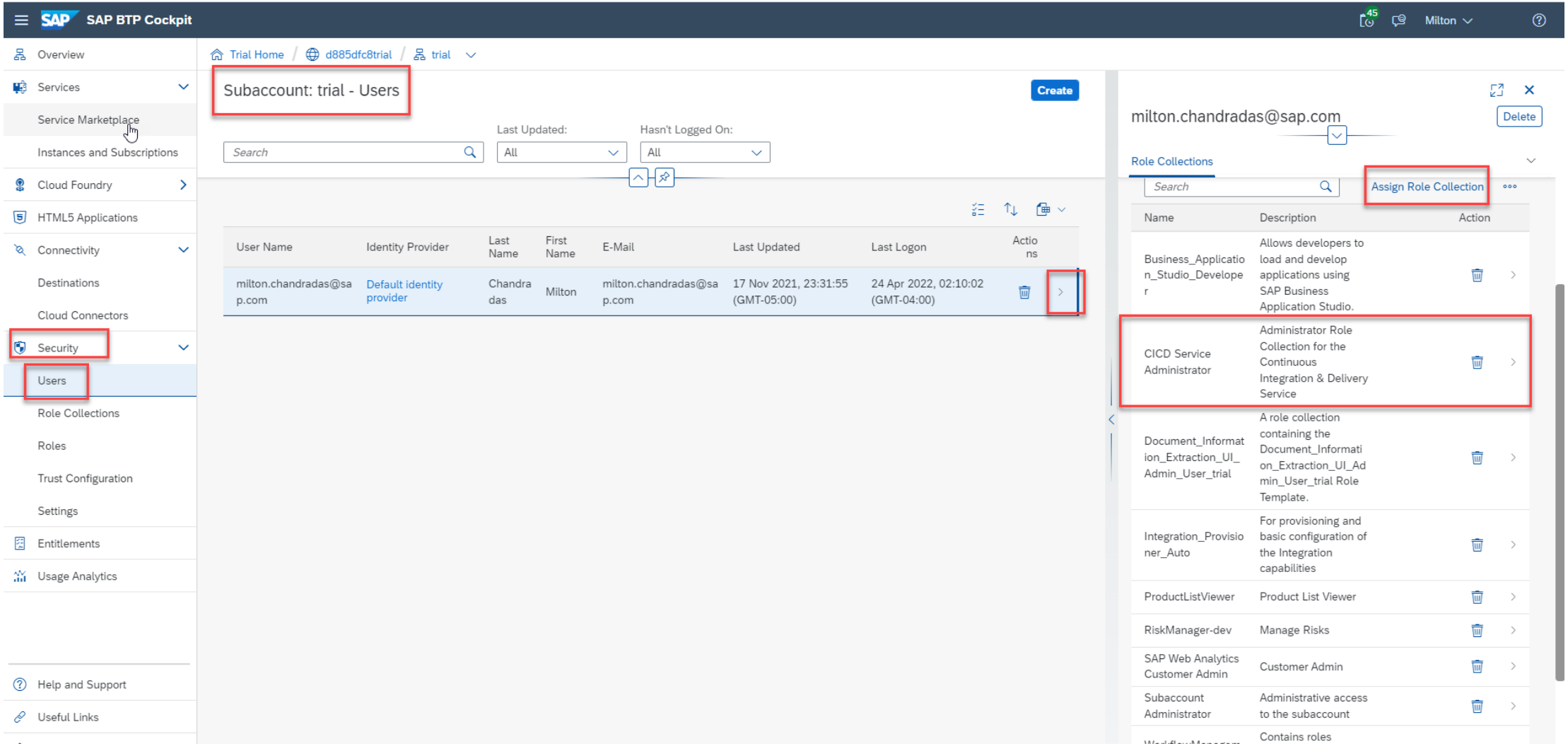
- Role has many Scopes

- Role-Collections contain 1 or more Roles
- Role-Collections can be assigned to a User

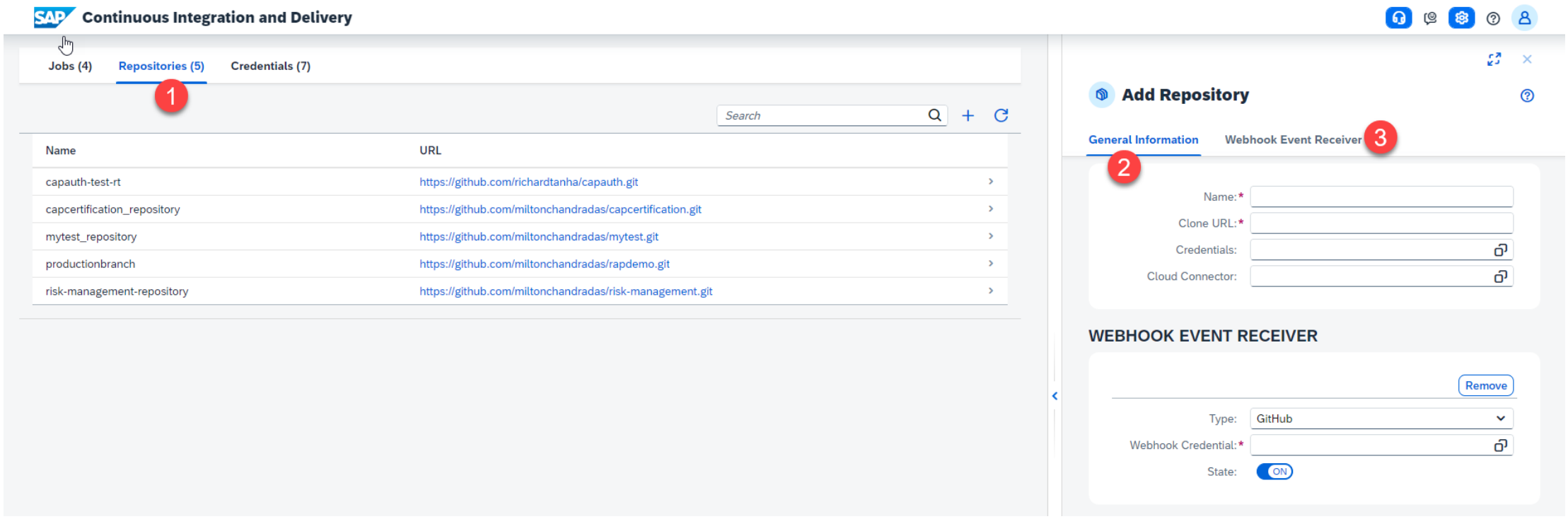
Key Summary Points – Unit 7



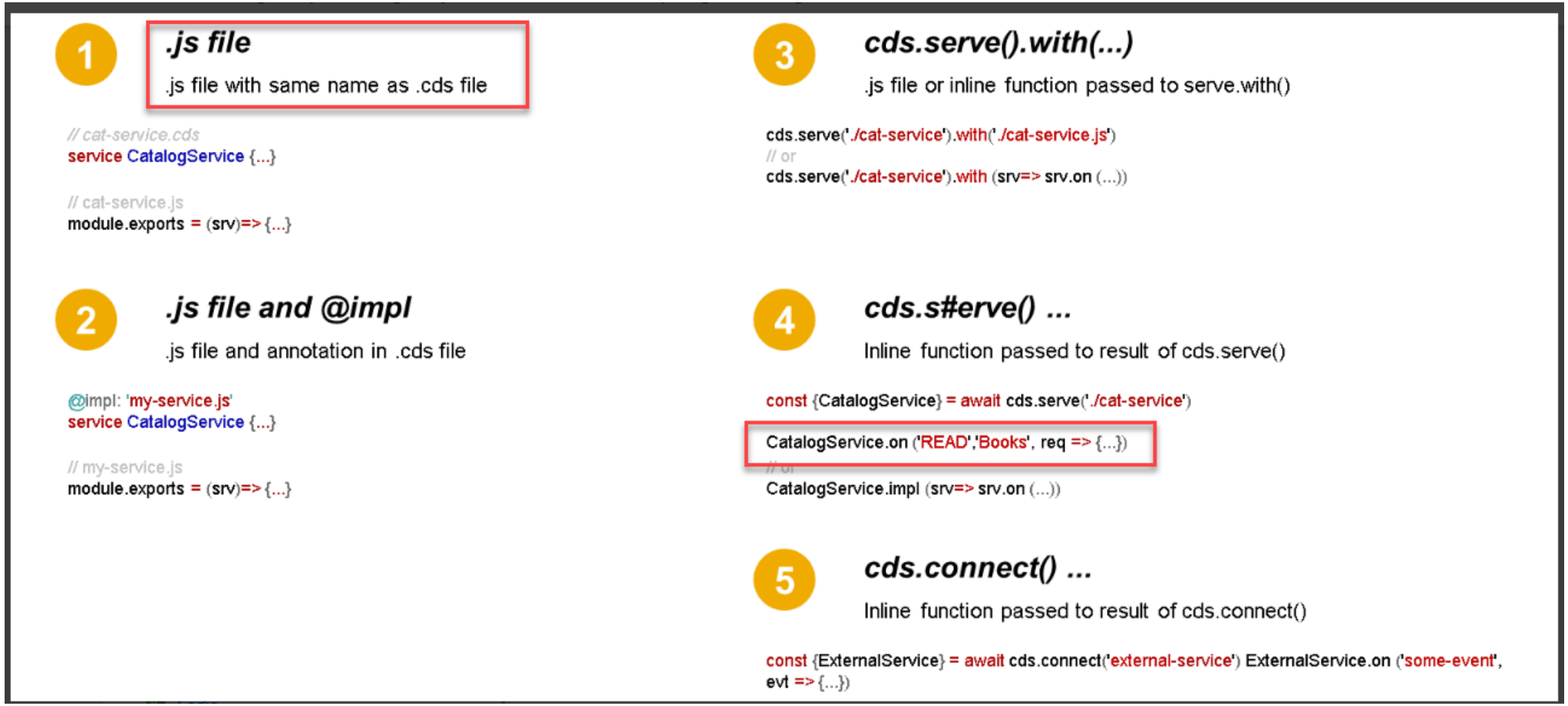
Key Summary Points – Unit 6



CI CD


Continuous Integration and Delivery
Jobs (4)
Repositories (5)
Credentials (7)
1
Search
Name
URL
capauth-test-rt
https://github.com/richardtanha/capauth.git
capcertification_repository
https://github.com/miltonchandradas/capcertification.git
mytest_repository
https://github.com/miltonchandradas/mytest.git
productionbranch
https://github.com/miltonchandradas/rapdemo.git
risk-management-repository
https://github.com/miltonchandradas/risk-management.git
Add Repository
General Information
Webhook Event Receiver
2
3
Name:*
Clone URL:*
Credentials:
Cloud Connector:
WEBHOOK EVENT RECEIVER
Remove
Type:
GitHub
Webhook Credential:*
State:
ON

Event handlers



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.
- [x] Identify users by e-mail, userId, first and last name.
- [x] 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?

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

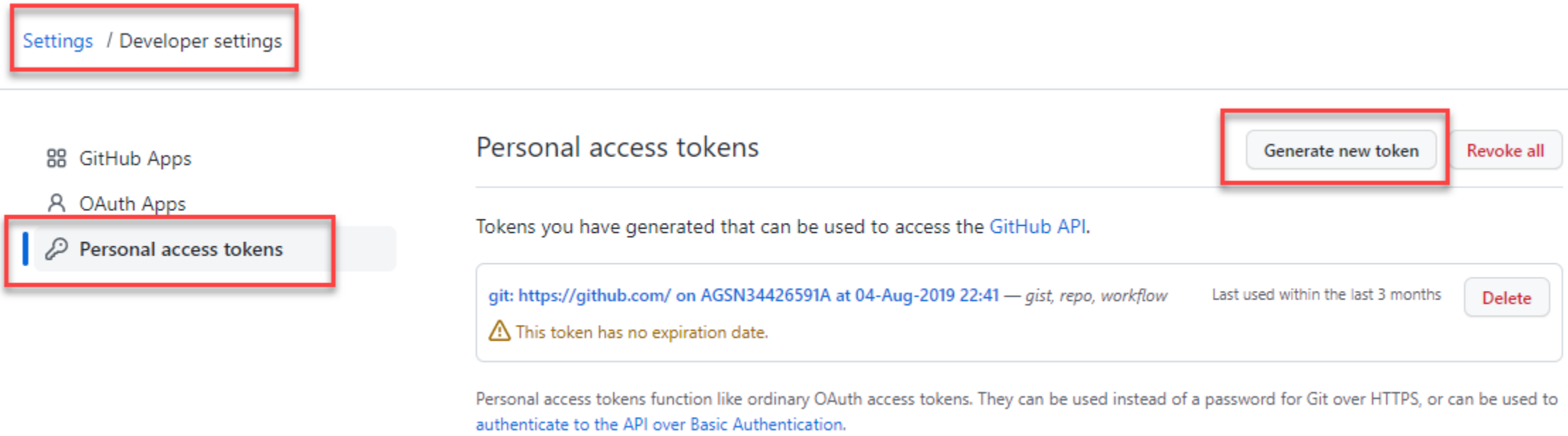
`xs-security.json` – Application Security Descriptor

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



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.

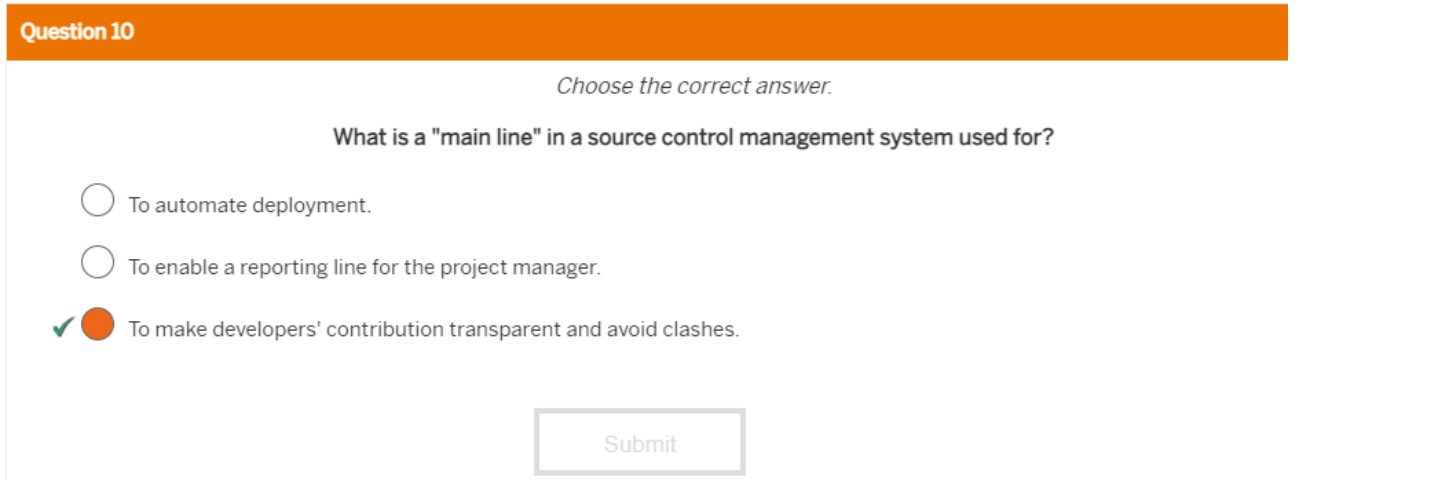
What does the source code management system use to trigger the CI server?

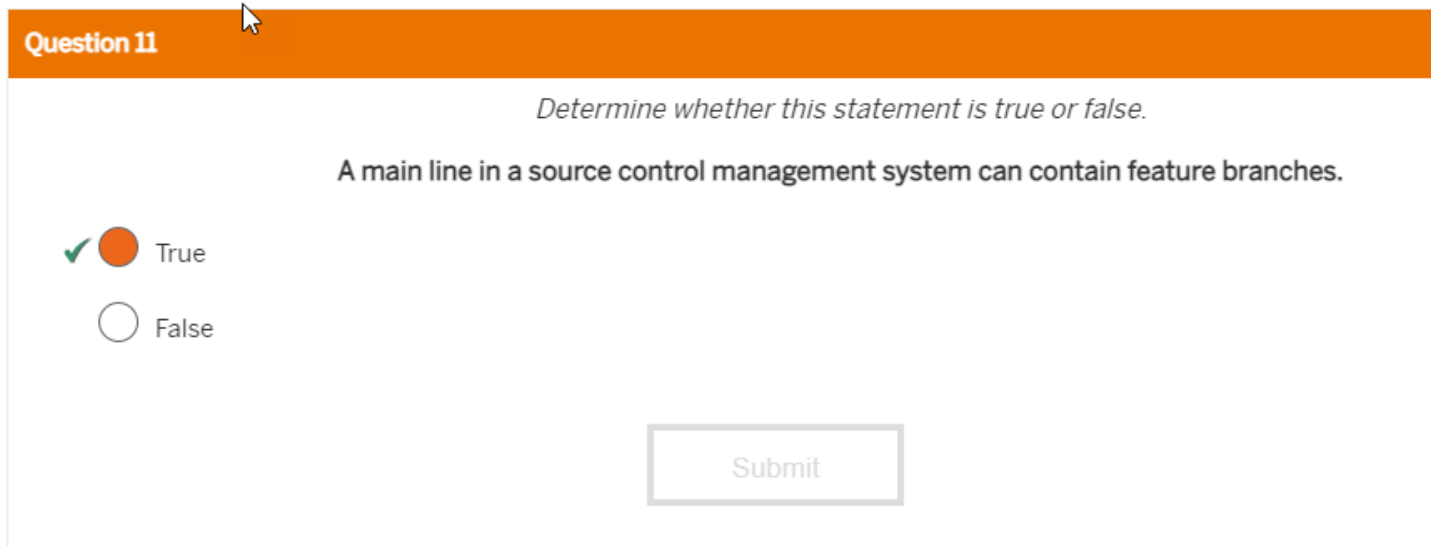
- ✔ Webhooks
- Web services
- HTTP PUT requests

Feedback

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

Key Summary Points – Unit 7





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

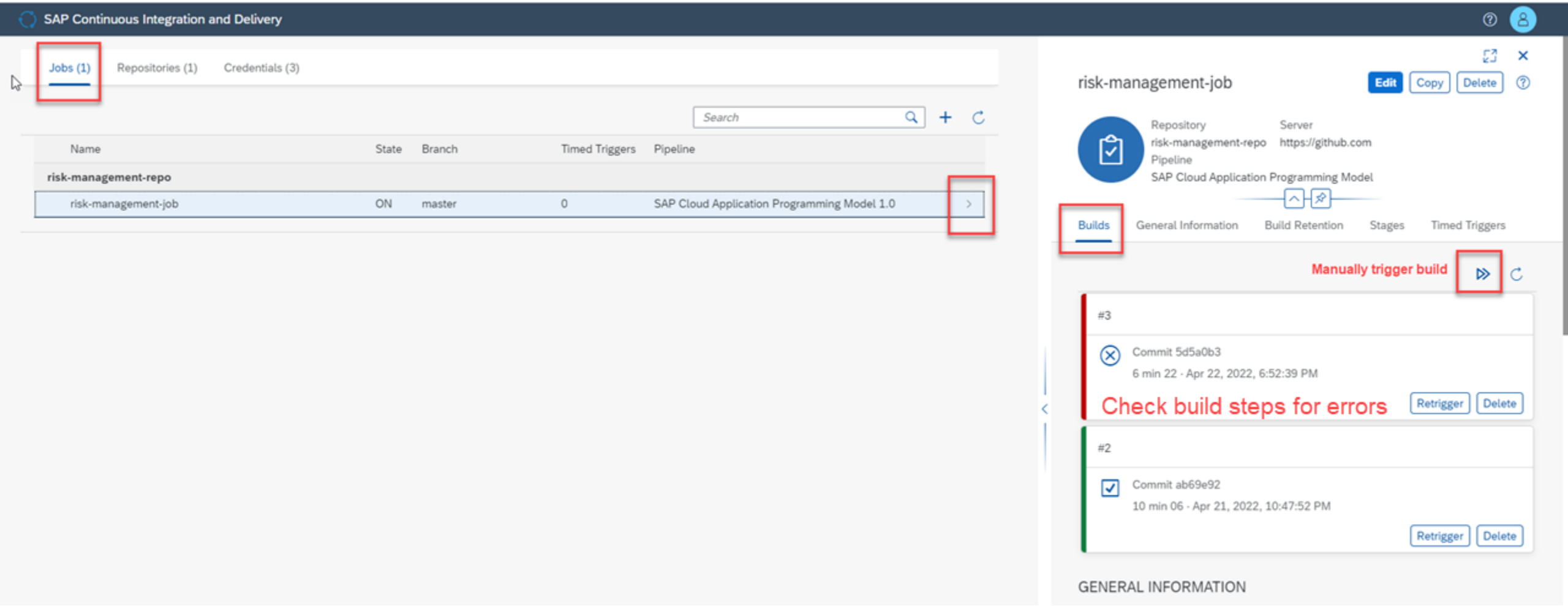
Question 15

Choose the correct answer.

What kind of request does the webhook send?

- ○ GET
- ○ PUT
- ✓ ● POST

Key Summary Points – Unit 7



INSTALLATION & GUIS

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 review 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

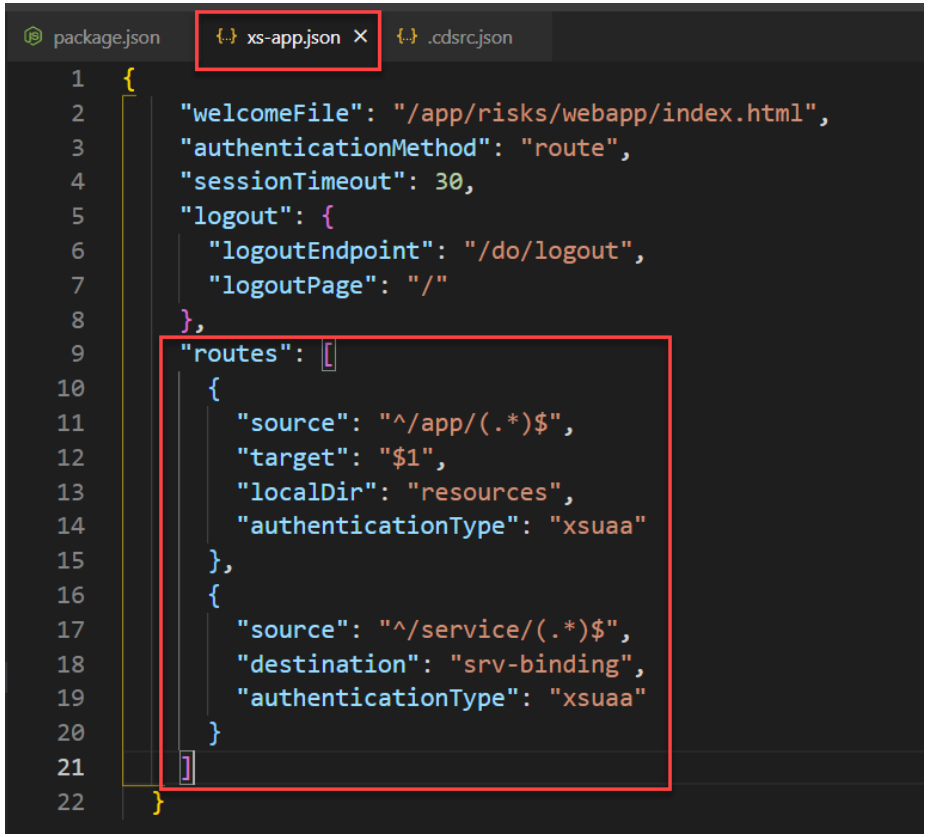
Isolating work in branches, changing context, and integrating changes

`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

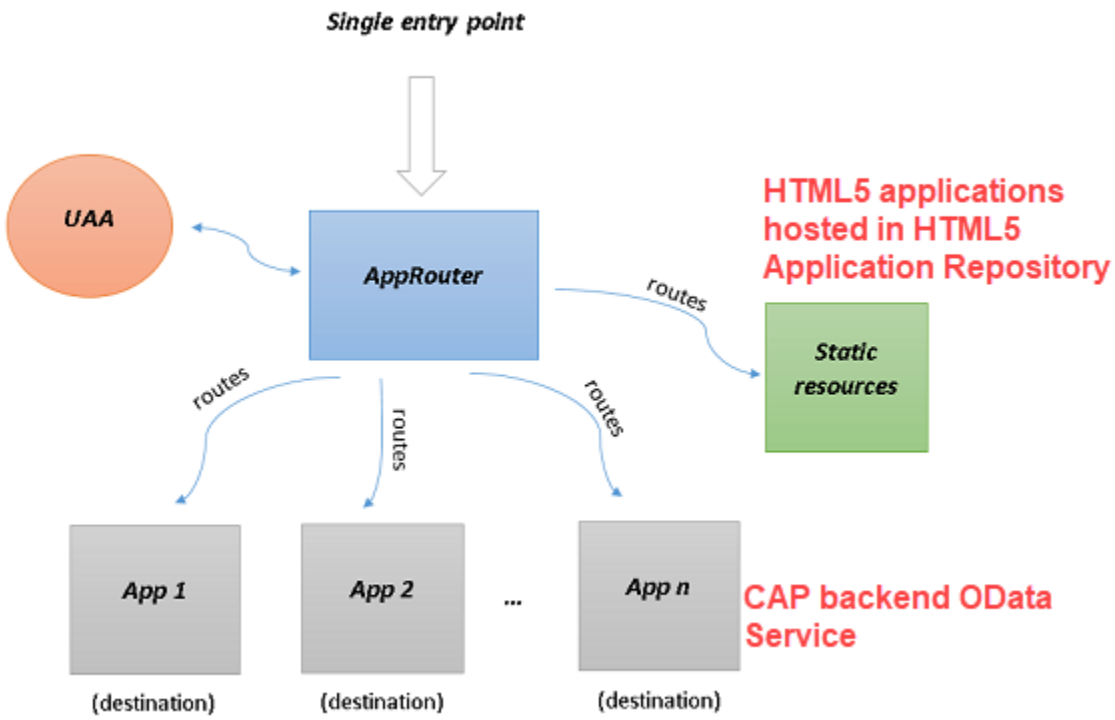
Key Summary Points – Unit 6

xs-app.json – Application Router Configuration

File contains configuration information used by the application router



Key Summary Points – Unit 6



AppRouter

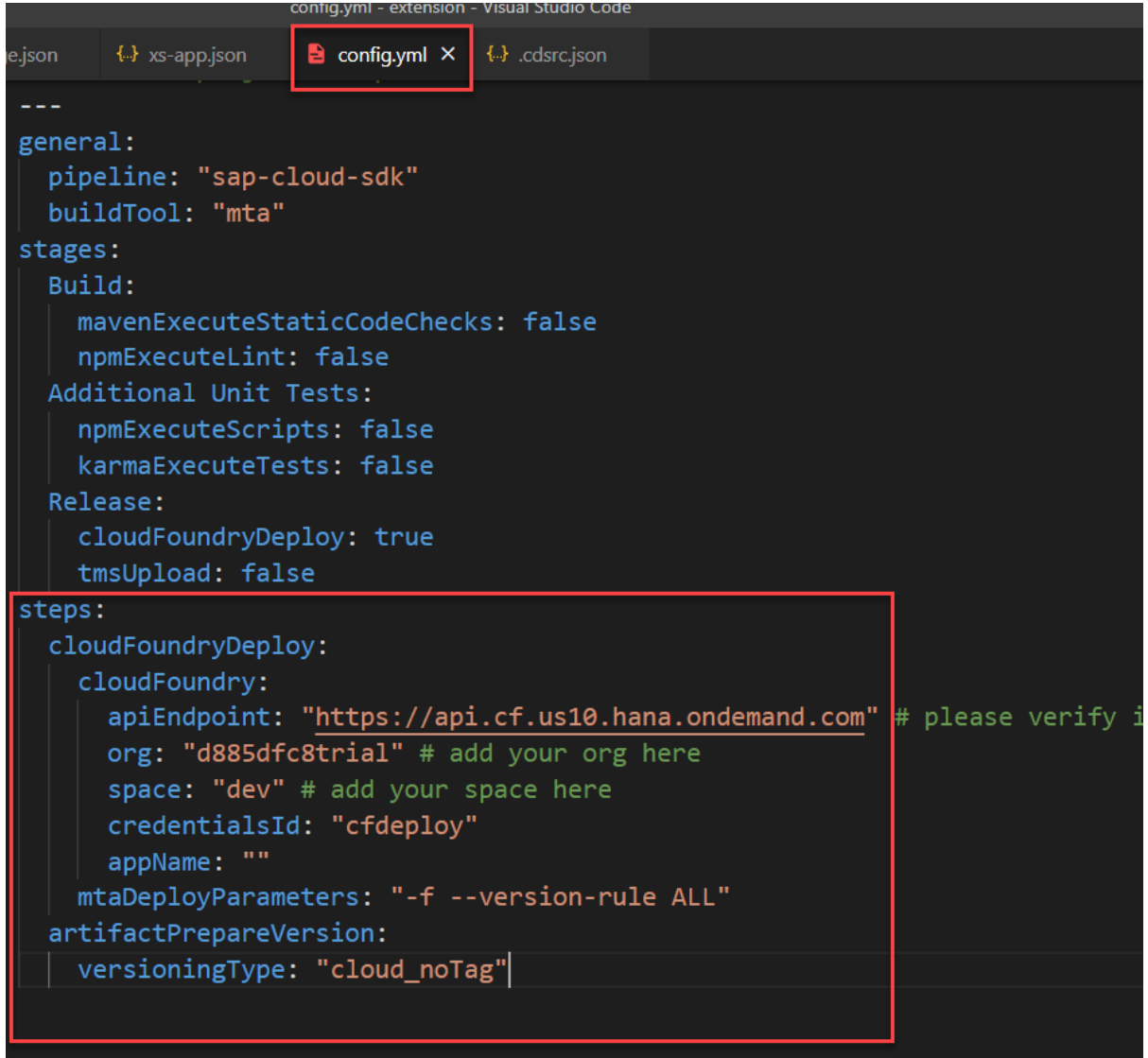
- Routes request from browser to CAP Service
- Routes request from browser to provider of UI sources
- Ensures authenticated and authorized users get token from XSUAA Service and forwards it to CAP Service

Key Summary Points – Unit 7

cds add pipeline

New Files

- Jenkinsfile
- .pipeline/config.yml



Key Summary Points – Unit 3

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

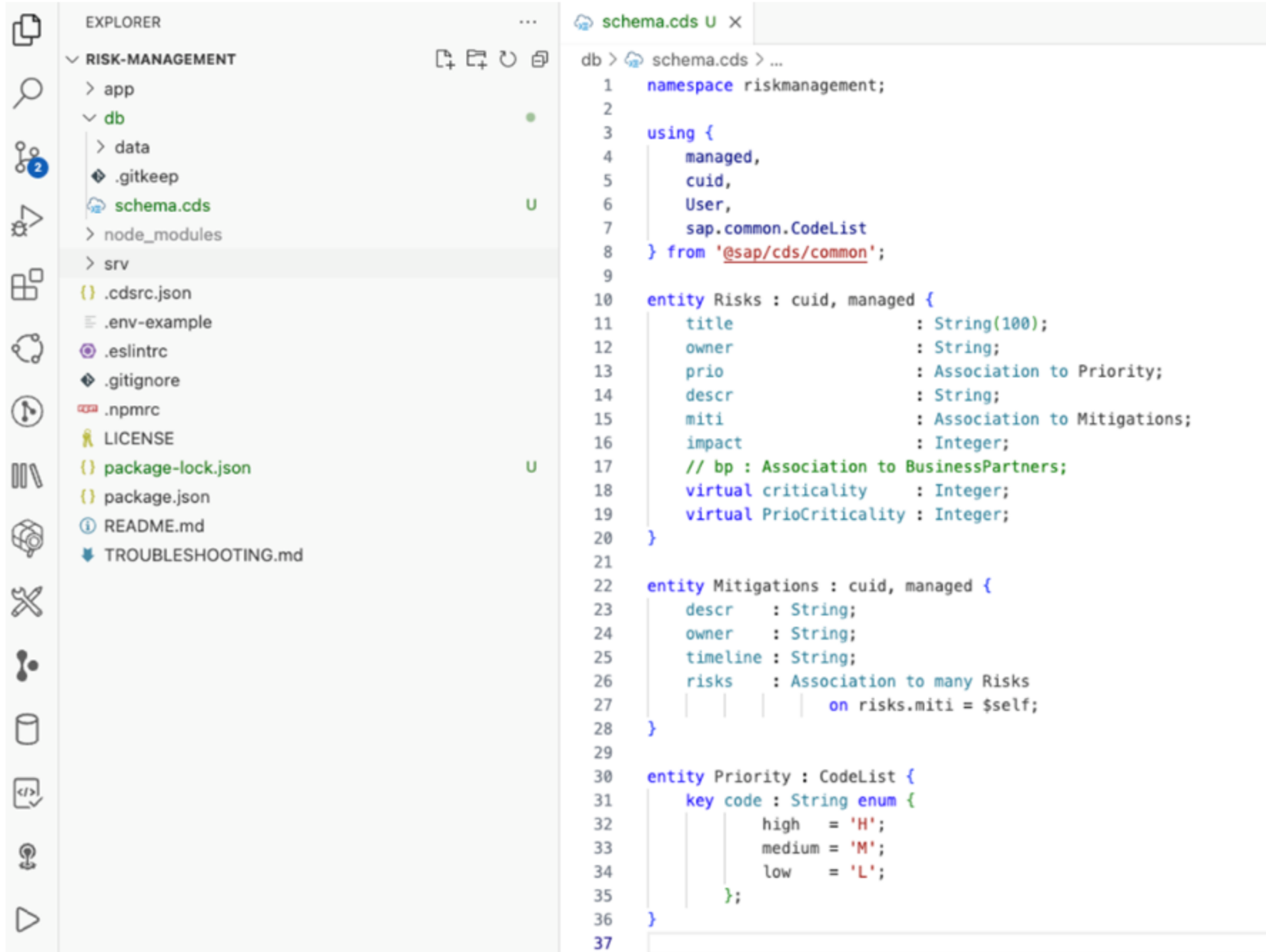
Use these methods to collect messages or error and return them in the request response to the caller. The method variants reflect different severity levels, use them as follows:

Variants

Method	*Collected in*	*Typical UI*	*Severity*
`req.notify`	`req.messages`	*Toasters*	*1*
`req.info`	`req.messages`	*Dialog*	*2*
`req.warn`	`req.messages`	*Dialog*	*3*
`req.error`	`req.error`	*Dialog*	*4*

Note: messages with severity < 4 a 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...
```


EXPLORER
RISK-MANAGEMENT
app
db
data
.gitkeep
schema.cds U
node_modules
srv
.cdsrc.json
.env-example
.eslintrc
.gitignore
.npmrc
LICENSE
package-lock.json U
package.json
README.md
TROUBLESHOOTING.md
schema.cds U
db > schema.cds > ...
1 namespace riskmanagement;
2
3 using {
4 managed,
5 cuid,
6 User,
7 sap.common.CodeList
8 } from '@sap/cds/common';
9
10 entity Risks : cuid, managed {
11 title : String(100);
12 owner : String;
13 prio : Association to Priority;
14 descr : String;
15 miti : Association to Mitigations;
16 impact : Integer;
17 // bp : Association to BusinessPartners;
18 virtual criticality : Integer;
19 virtual PrioCriticality : Integer;
20 }
21
22 entity Mitigations : cuid, managed {
23 descr : String;
24 owner : String;
25 timeline : String;
26 risks : Association to many Risks
27 on risks.miti = $self;
28 }
29
30 entity Priority : CodeList {
31 key code : String enum {
32 high = 'H';
33 medium = 'M';
34 low = 'L';
35 };
36 }
37

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.
- B Changing the field's position in an existing screen.
- ✓ Extending the existing application to include new functions.
- D 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

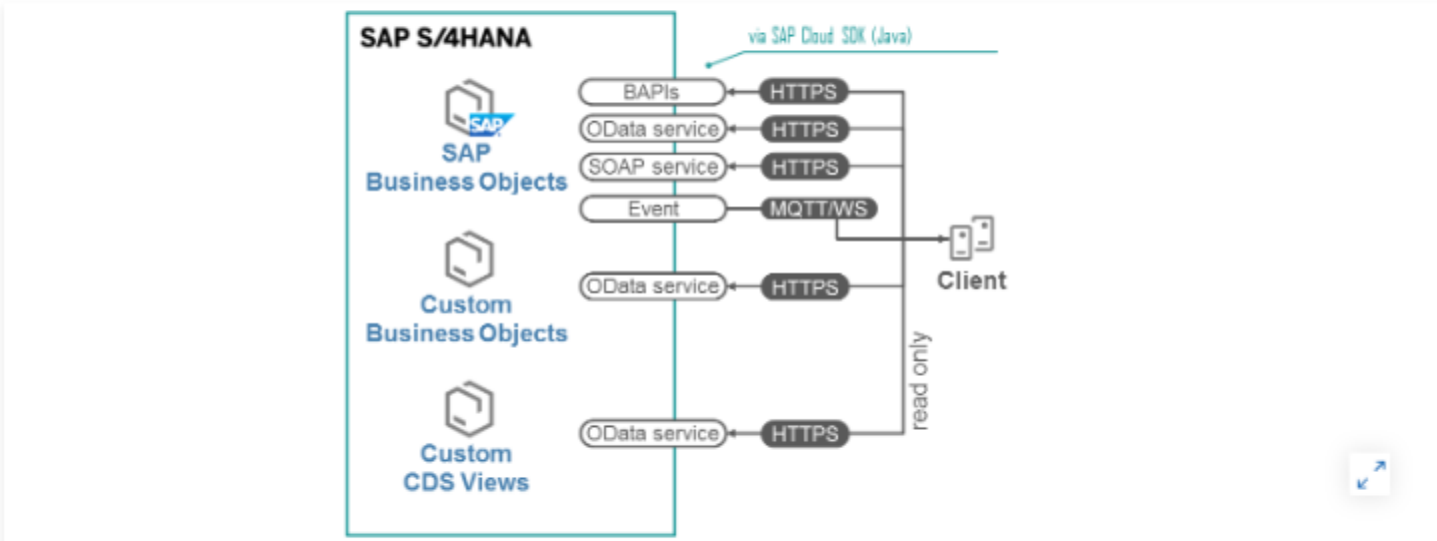
If you already have an existing Java application that was built without using the SAP CAP Model and you want to add more features to it. Here again, you can make use of the SAP Cloud SDK tool set.

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);
```