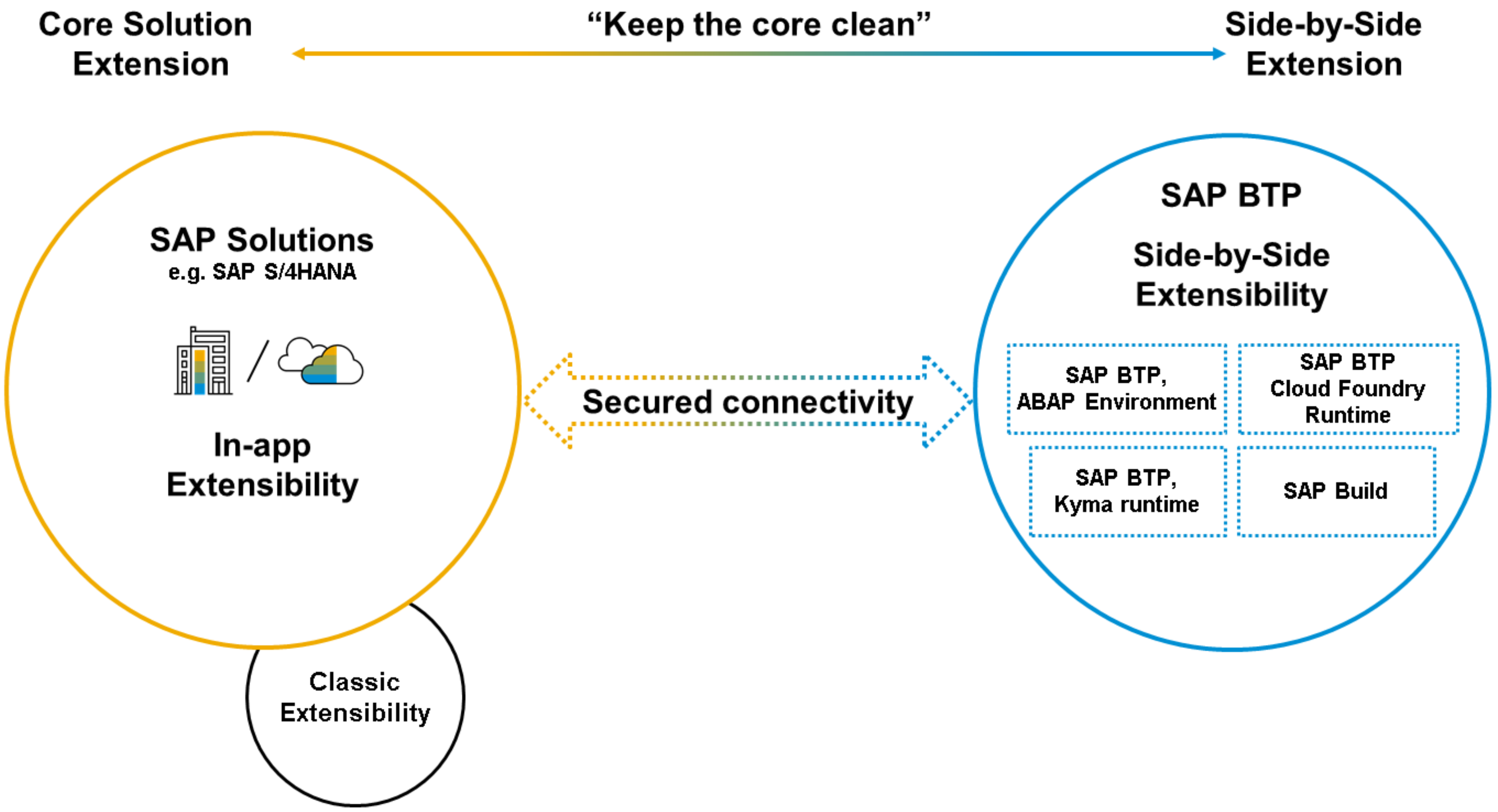
Side-by-Side Extensions on SAP BTP

Unit 1 – Exploring Side-by-Side Extensibility



Unit 1 – Extensibility Options on SAP BTP

Categorizing the skill set of SAP Cloud Developers

SAP Cloud Developers can specialize in **either ABAP or Non-ABAP** development.

Each with its unique set of tools and methodologies.

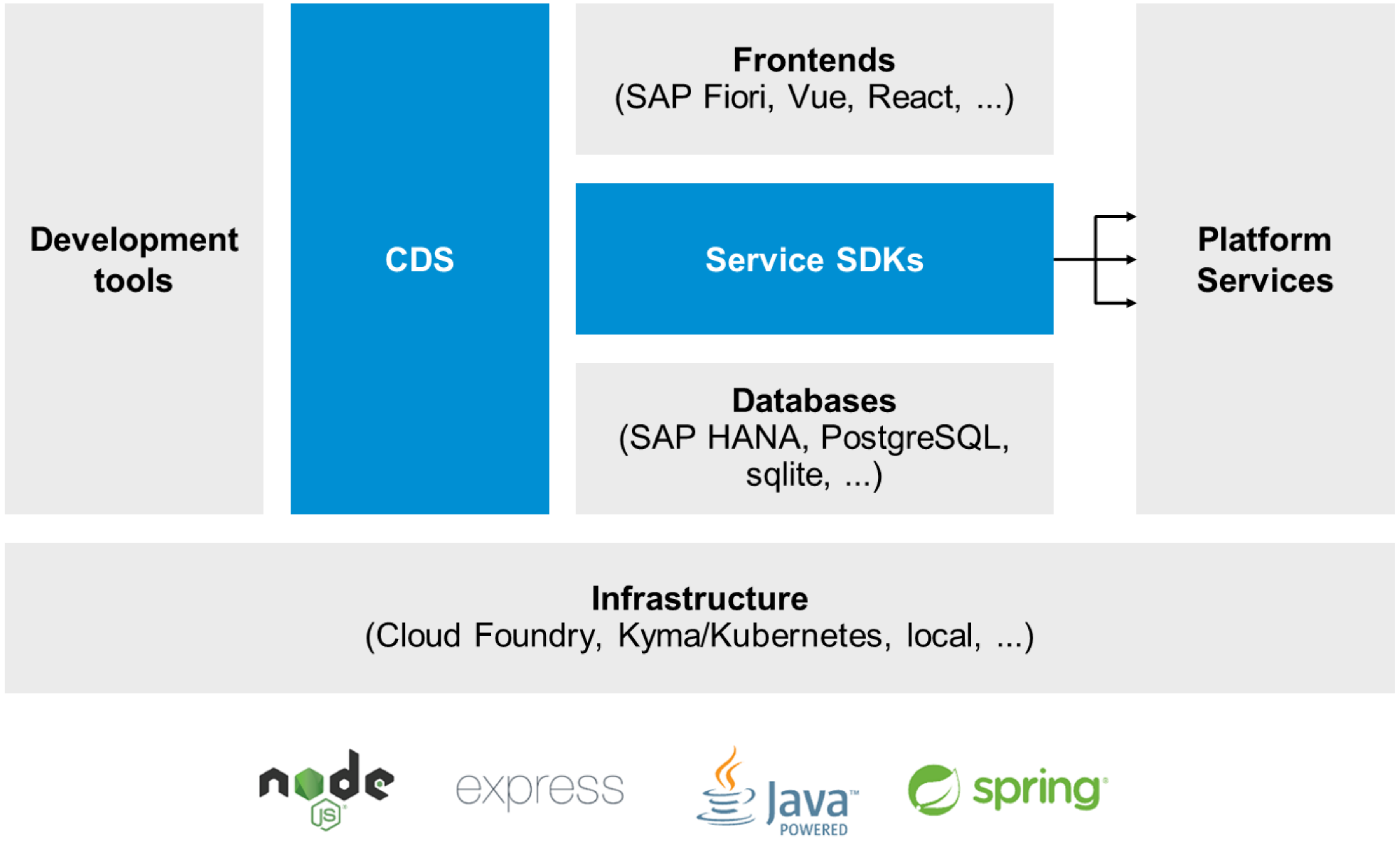
ABAP-based

- Utilizes ABAP RESTful Programming Model (RAP)
- Supports only ABAP (Cloud)
- Focuses on S/4 HANA extensions
- Development only to on-prem / ABAP Environment on SAP BTP

Non-ABAP-based

- Utilizes Cloud Application Programming Model (CAP) or SAP Cloud SDK
- Supports JavaScript, TypeScript, Java
- Target multi-microservice-based SaaS & SAP S/4 HANA extensions
- Development to Cloud Foundry / Kyma Runtime

Unit 1 – What is SAP CAP model ?


Development tools
CDS
Frontends (SAP Fiori, Vue, React, ...)
Service SDKs
Databases (SAP HANA, PostgreSQL, sqlite, ...)
Platform Services
Infrastructure (Cloud Foundry, Kyma/Kubernetes, local, ...)
node JS
express
Java POWERED
spring

Unit 2 – Introducing OData protocol

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

Unit 2 – Explaining JSON / YAML

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

Discovering End-to-End Use Case

Application Features

- OData V4 Service
- SAP Fiori Elements Application
- Seamless integration
  - External Services from SAP S/4HANA Cloud
  - Local Services from hdi instance
- Manual Deployment
  - mta.yml
- Security – Authentication and Authorization
  - Local development
  - CF deployment
- CI / CD

Unit 2 – Exercise (Data Modeling)

```
git clone https://github.com/miltonchandradas/risk-management.git

cds init <Name of Project>

git checkout unit2_setup (Use tab for branch name)
```

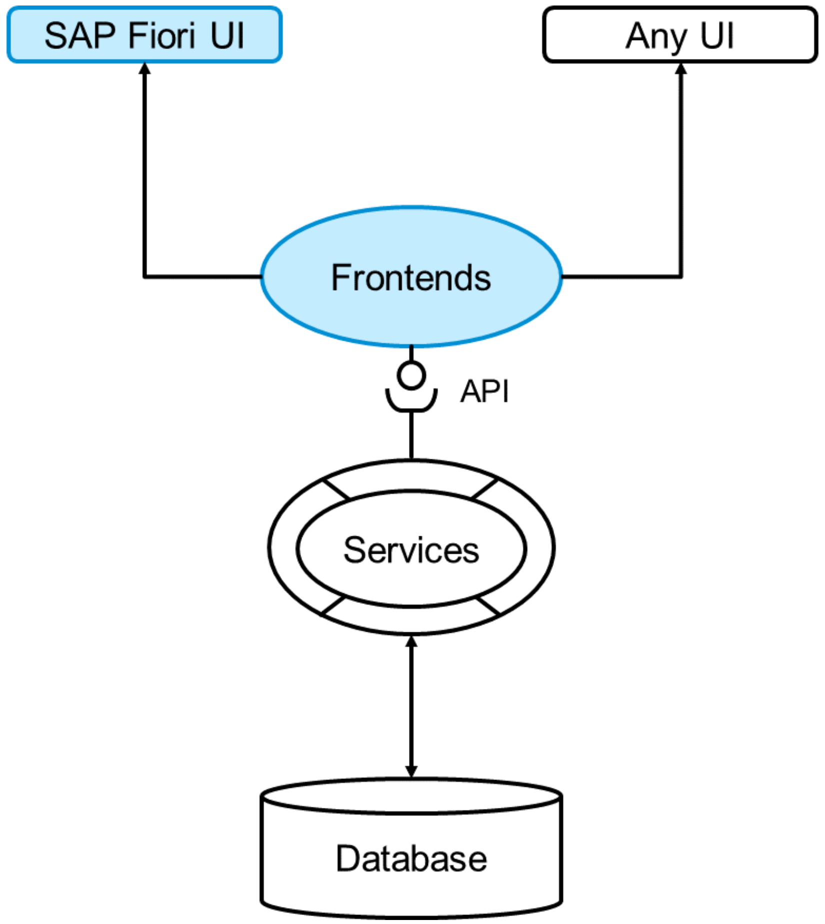
Folders created

- app
- db
- srv

Files created

- package.json

Unit 3 – Frontend Capabilities



Unit 3 – Exercise (User Interface)

```
git checkout unit3_ui (Use tab for branch name)
```

```
SAP Fiori Elements applications
UI can be influenced by OData annotations
```

Unit 4 – Custom Logic

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() ...*

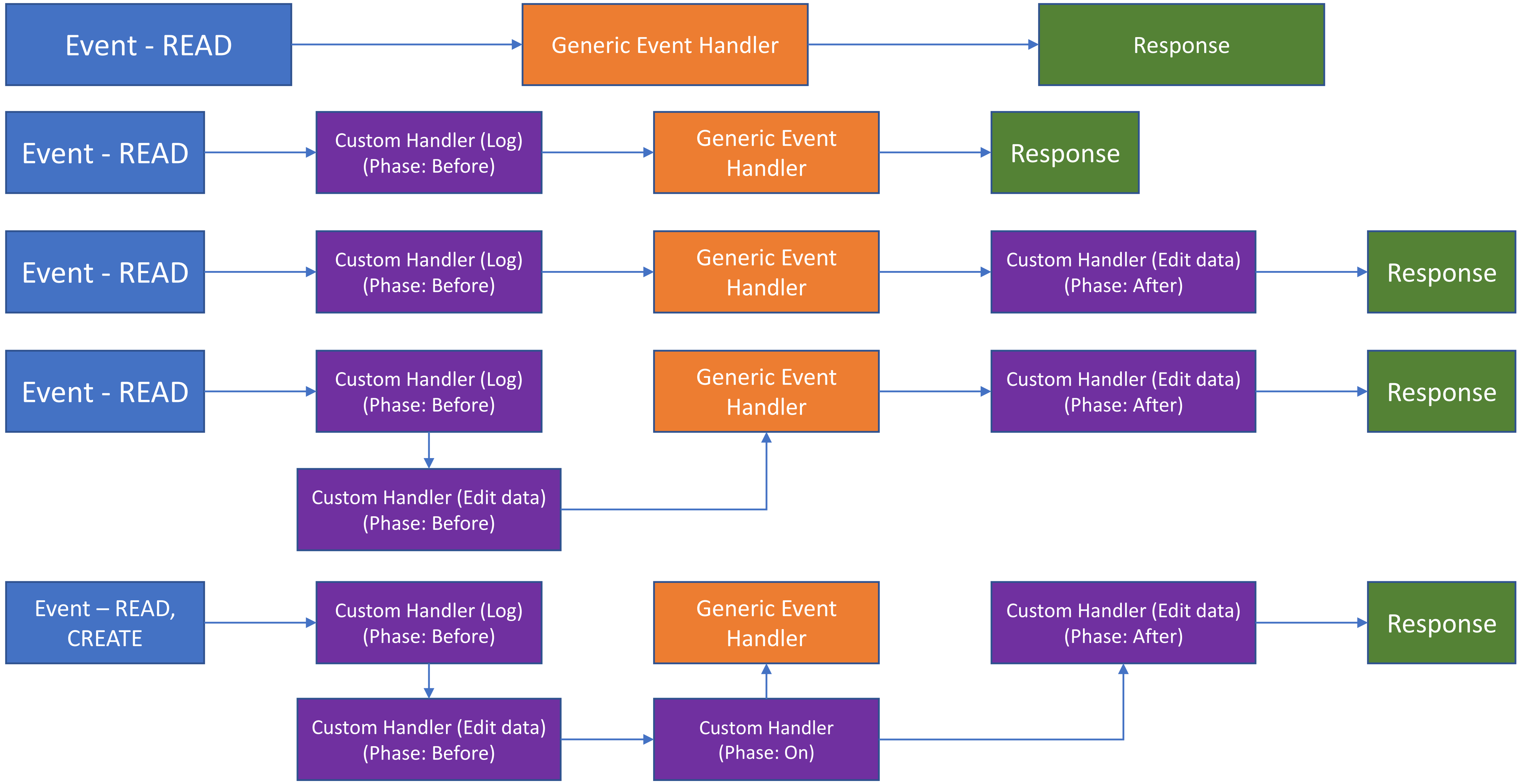
Inline function passed to result of cds.serve()

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

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

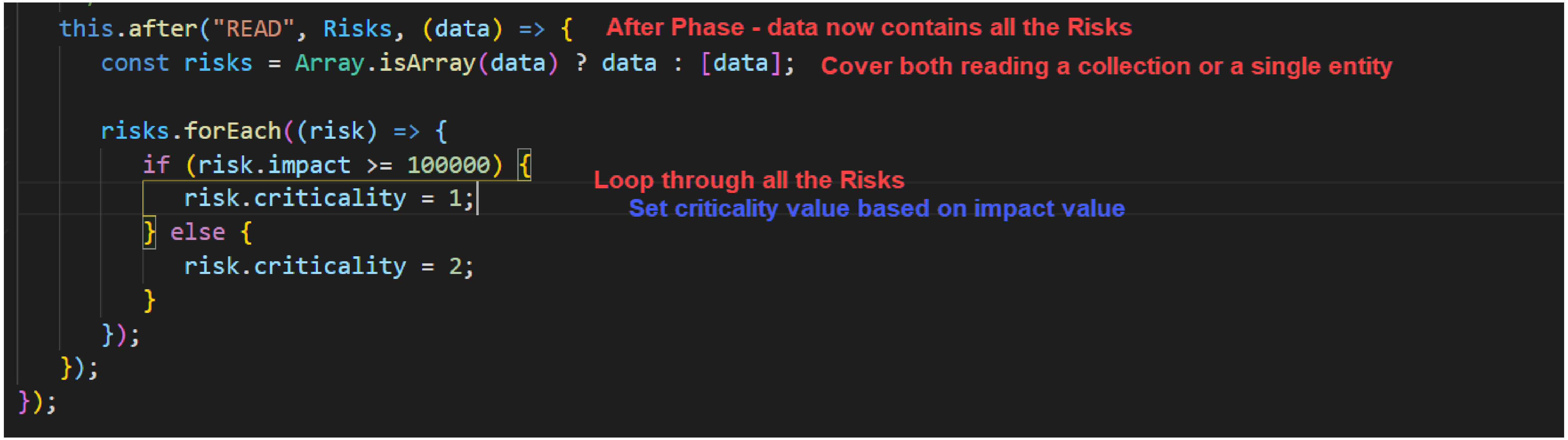
5 *cds.connect() ...*

Inline function passed to result of cds.connect()

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


Event - READ
Generic Event Handler
Response
Event - READ
Custom Handler (Log) (Phase: Before)
Generic Event Handler
Response
Event - READ
Custom Handler (Log) (Phase: Before)
Generic Event Handler
Custom Handler (Edit data) (Phase: After)
Response
Event - READ
Custom Handler (Log) (Phase: Before)
Generic Event Handler
Custom Handler (Edit data) (Phase: After)
Response
Custom Handler (Edit data) (Phase: Before)
Event – READ, CREATE
Custom Handler (Log) (Phase: Before)
Generic Event Handler
Custom Handler (Edit data) (Phase: After)
Response
Custom Handler (Edit data) (Phase: Before)
Custom Handler (Phase: On)

Step 6 – Custom Event Handling



```
    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;
            }
        });
    });
});
```

After Phase - data now contains all the Risks

Cover both reading a collection or a single entity

Loop through all the Risks

Set criticality value based on impact value

Error Handling

Guidelines:

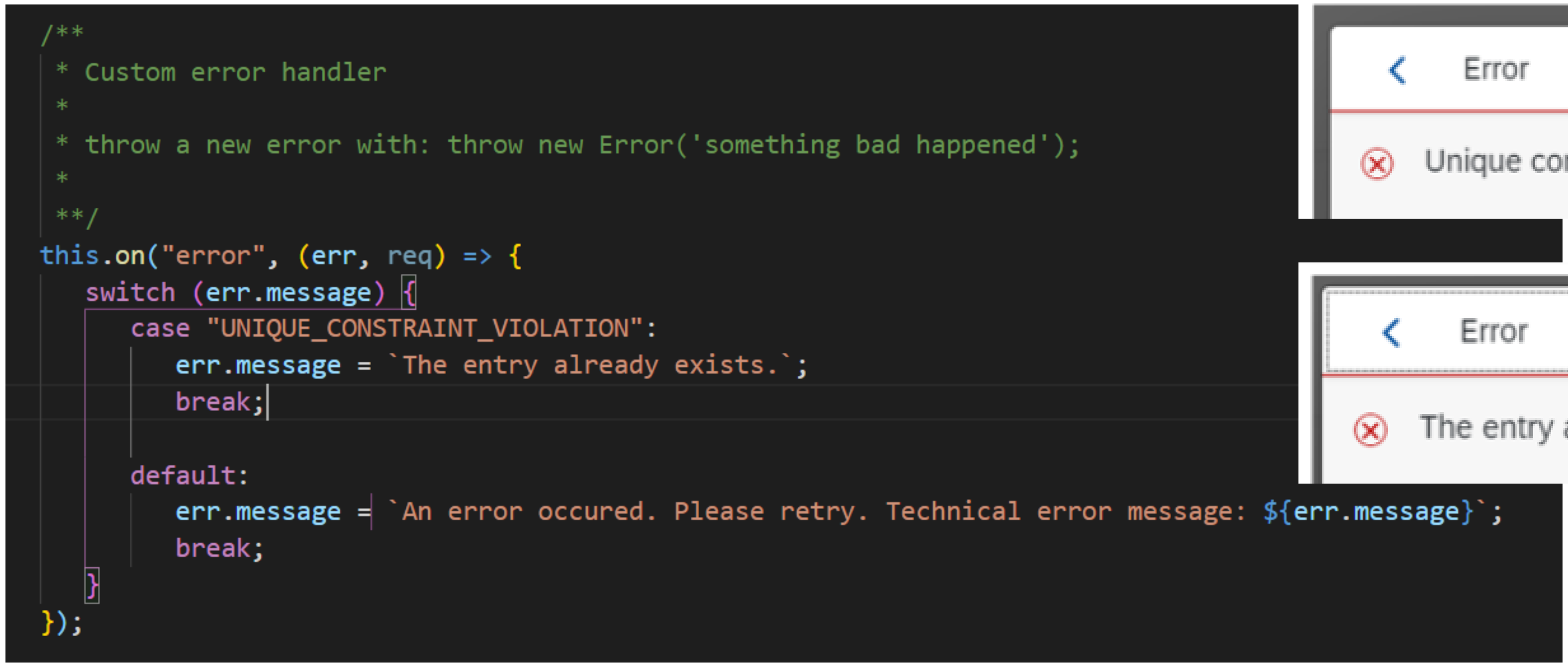
Fail loudly. Do not hide errors and continue silently.

Log unexpected errors

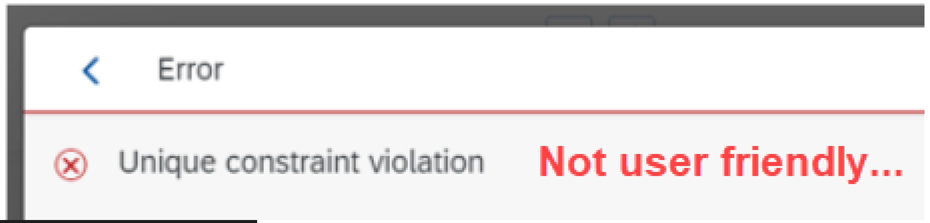
Don’t catch errors you can’t handle

Use try / catch blocks only when necessary

Error Handling

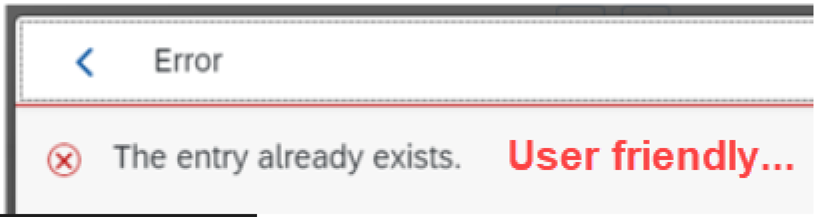


```
/**
 * Custom error handler
 *
 * throw a new error with: throw new Error('something bad happened');
 *
 **/
this.on("error", (err, req) => {
   switch (err.message) {
      case "UNIQUE_CONSTRAINT_VIOLATION":
         err.message = `The entry already exists.`;
         break;

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



Error

Unique constraint violation Not user friendly...



Error

The entry already exists. User friendly...

Register an error handler in your service implementation

- Provide meaningful error message to end user

Unit 4 – Exercise (Custom logic)

```
git checkout unit4_customlogic (Use tab for branch name)
```

`Custom code to CAP service to implement conditional formatting`

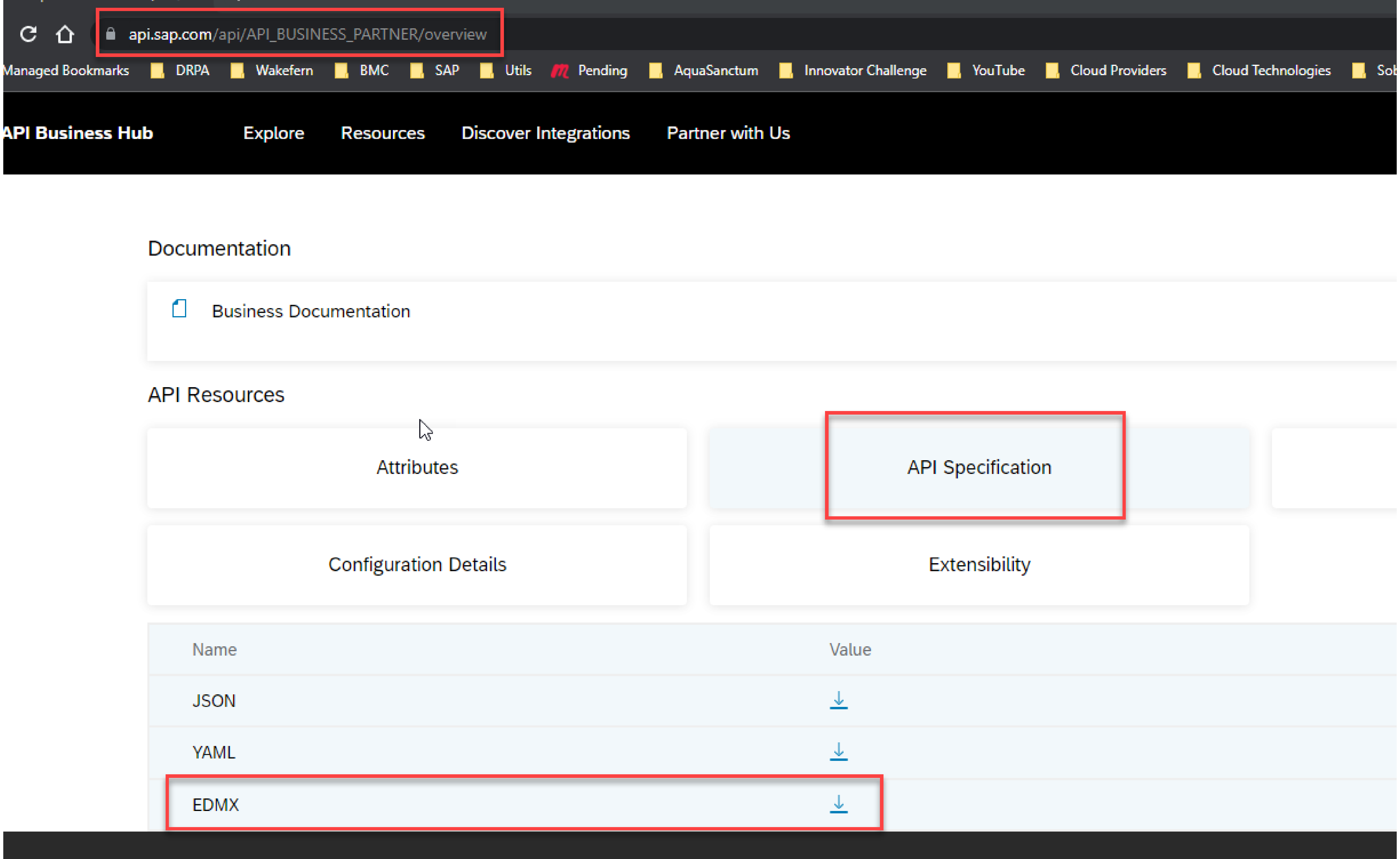
Unit 5 – Support for External API

SAP Business Accelerator Hub

https://api.sap.com

SAP S/4HANA Cloud – Business Partner

https://api.sap.com/api/API_BUSINESS_PARTNER/overview

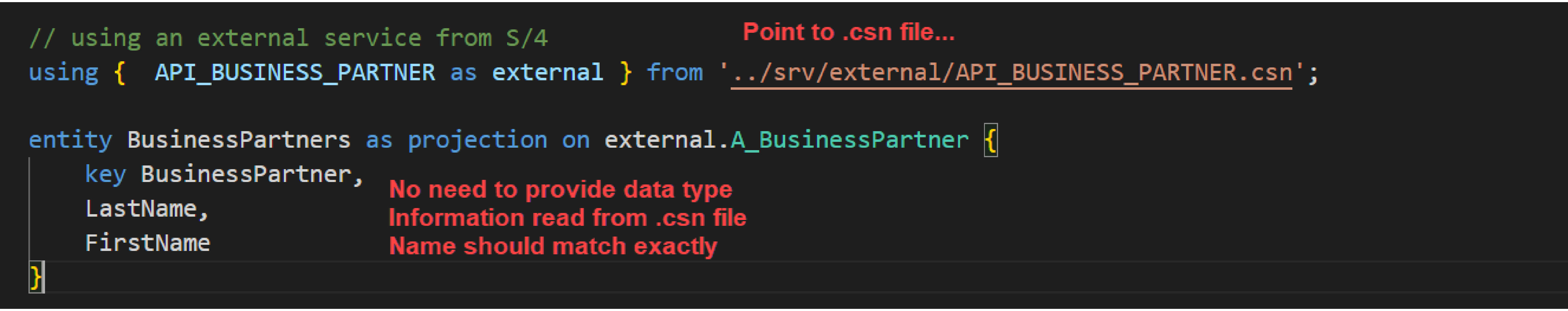

api.sap.com/api/API_BUSINESS_PARTNER/overview

Documentation

Business Documentation

API Resources

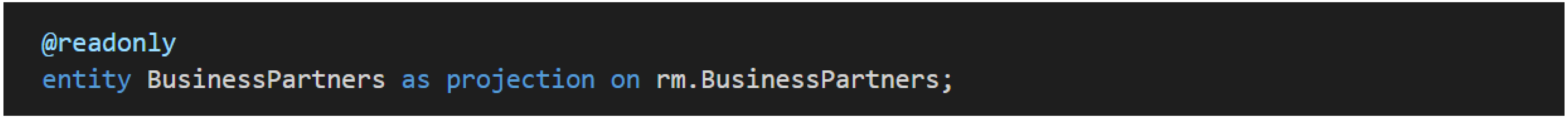
Name	Value
JSON	⤓
YAML	⤓
EDMX	⤓



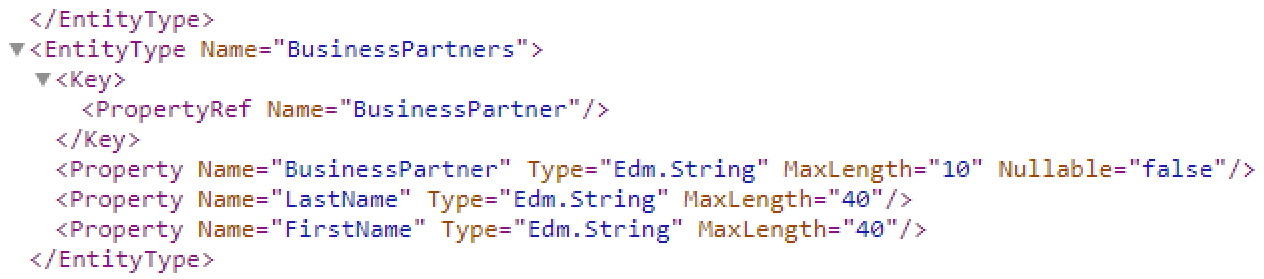
```
// using an external service from S/4
using {  API_BUSINESS_PARTNER as external } from '../srv/external/API_BUSINESS_PARTNER.csn';

entity BusinessPartners as projection on external.A_BusinessPartner {
    key BusinessPartner,
    LastName,
    FirstName
}
```

Point to .csn file...

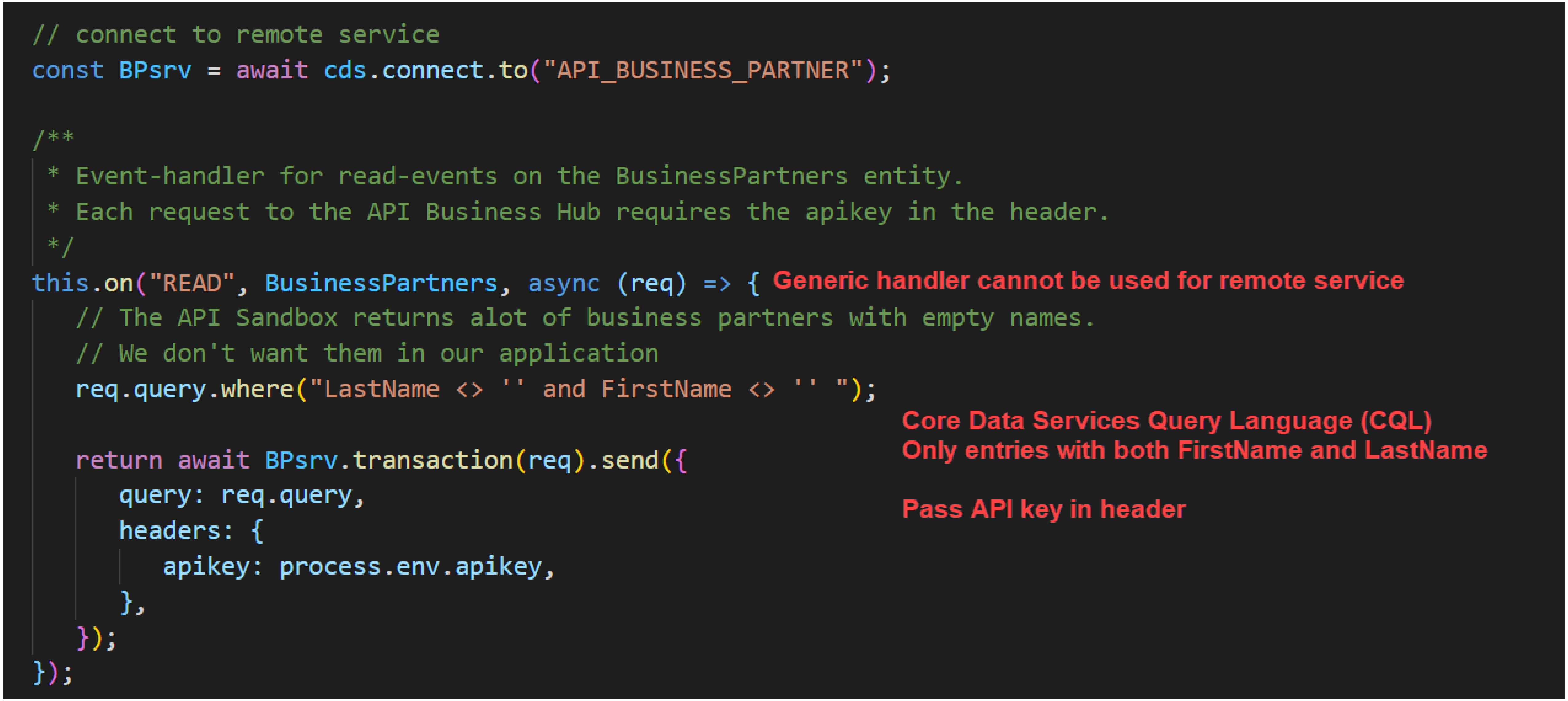
No need to provide data type
Information read from .csn file
Name should match exactly



```
@readonly
entity BusinessPartners as projection on rm.BusinessPartners;
```



```
</EntityType>
▼<EntityType Name="BusinessPartners">
 ▼<Key>
    <PropertyRef Name="BusinessPartner"/>
  </Key>
  <Property Name="BusinessPartner" Type="Edm.String" MaxLength="10" Nullable="false"/>
  <Property Name="LastName" Type="Edm.String" MaxLength="40"/>
  <Property Name="FirstName" Type="Edm.String" MaxLength="40"/>
</EntityType>
```



```
// connect to remote service
const BPsrv = await cds.connect.to("API_BUSINESS_PARTNER");

/**
 * Event-handler for read-events on the BusinessPartners entity.
 * Each request to the API Business Hub requires the apikey in the header.
 */
this.on("READ", BusinessPartners, async (req) => {
  // The API Sandbox returns alot of business partners with empty names.
  // We don't want them in our application
  req.query.where("LastName <> '' and FirstName <> '' ");

  return await BPsrv.transaction(req).send({
    query: req.query,
    headers: {
      apikey: process.env.apikey,
    },
  });
});
```

Generic handler cannot be used for remote service

Core Data Services Query Language (CQL)
Only entries with both FirstName and LastName

Pass API key in header

srv/risk-service.js

UI makes the following OData call…

GET
http://localhost:4004/service/risk/Risks?$expand=bp,miti($select=ID,IsActiveEntity,descr)

GET all Risks

- GET corresponding bp details for each Risk
- GET corresponding miti details for each Risk

Risks

?$expand=bp,miti($select=ID,IsActiveEntity,descr)

Generic handler cannot expand bp ➔ So we need to have custom handler to expand bp

srv/risk-service.js

Aim:

Read data from external data source along with Risks

Custom event handler: On, READ, Risks

Steps:

1. Remove expand bp clause from request query
2. Call the generic READ handler
3. Call the external data source for BP info

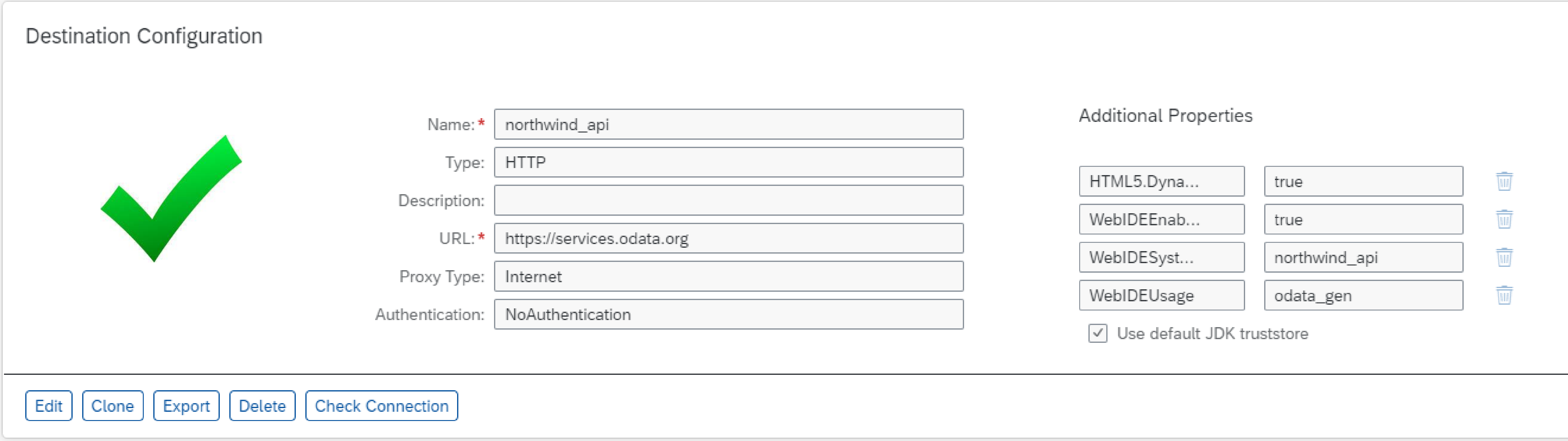
SAP BTP Connectivity

- Cloud Foundry environment
  - 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

Destination Service – CF

Works for any publicly available OData Service
https://services.odata.org/v2/northwind/northwind.svc/



Destination Configuration

Name:*	northwind_api
Type:	HTTP
Description:	
URL:*	https://services.odata.org
Proxy Type:	Internet
Authentication:	NoAuthentication

Additional Properties

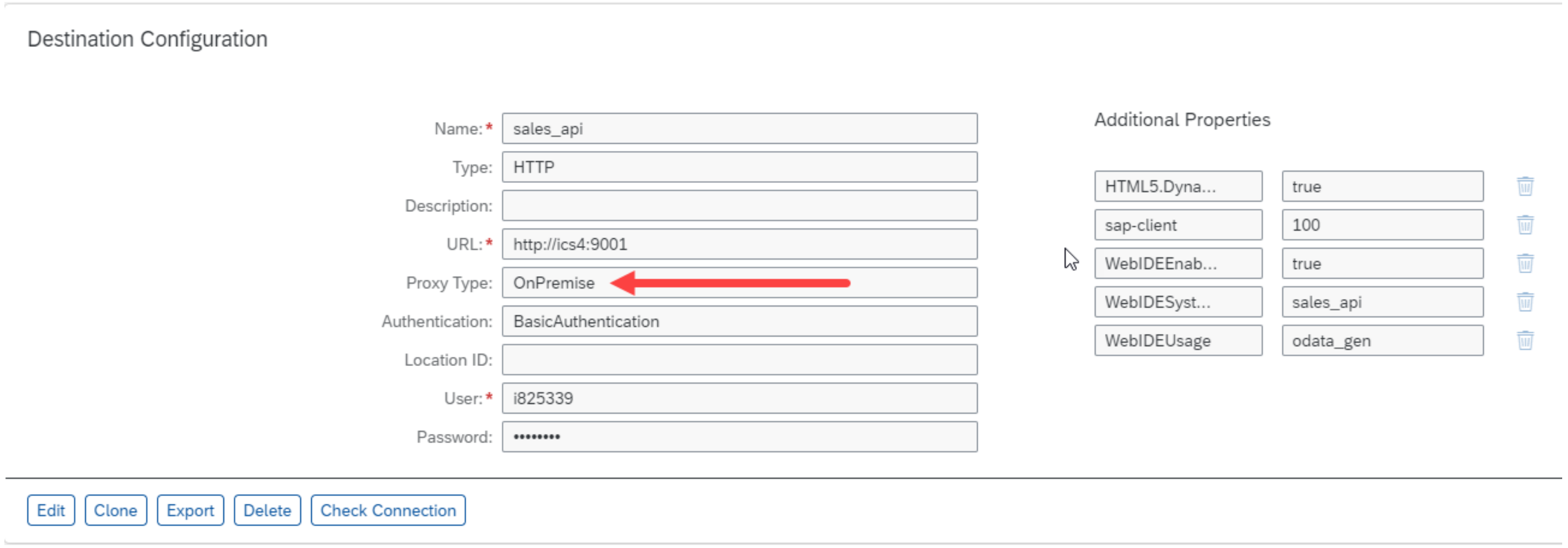
HTML5.Dyna...	true
WebIDEEnab...	true
WebIDESyst...	northwind_api
WebIDEUsage	odata_gen

[✓] Use default JDK truststore

Destination Service – CF

Scenarios:

1. Connecting to On-Premise OData Service
2. Retrieve connection details


Destination Configuration

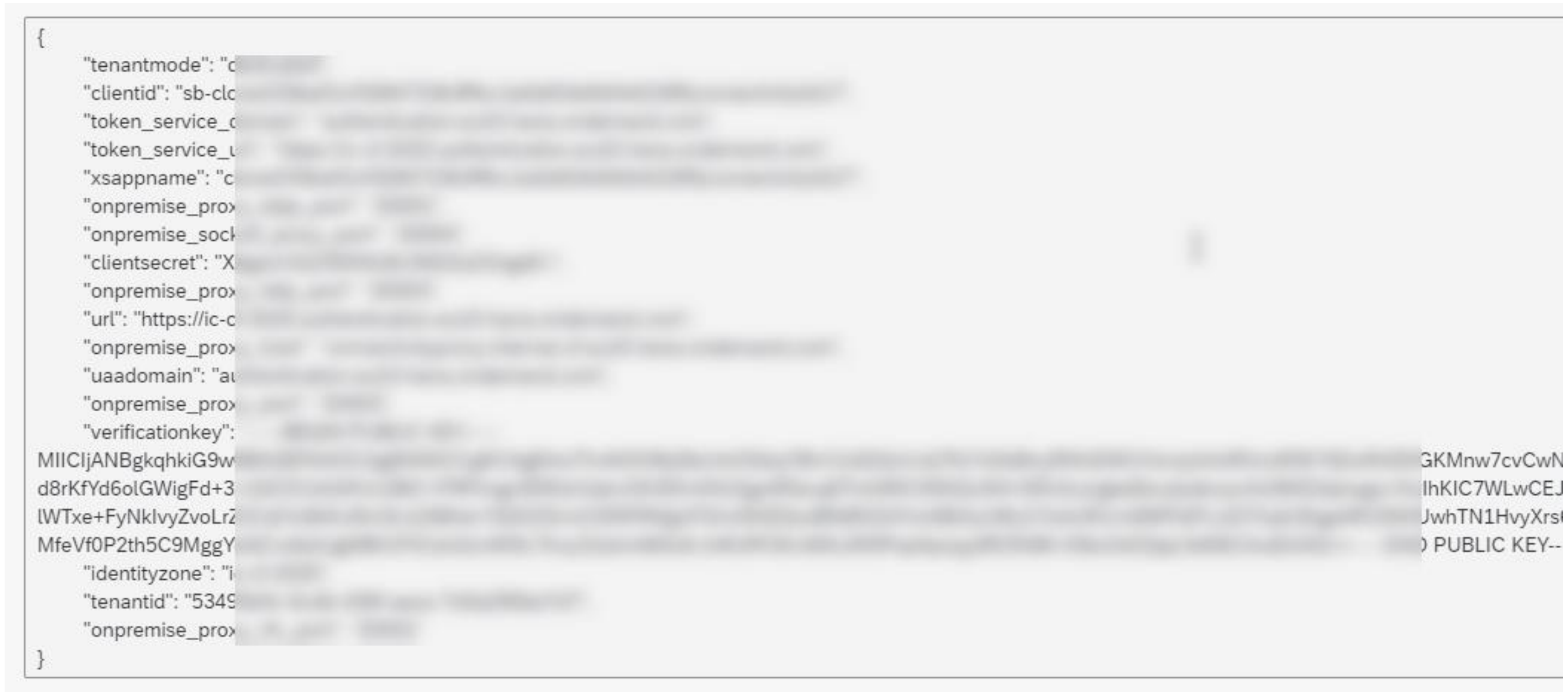
Name:*	sales_api
Type:	HTTP
Description:	
URL:*	http://ics4:9001
Proxy Type:	OnPremise
Authentication:	BasicAuthentication
Location ID:	
User:*	i825339
Password:	••••••••

Additional Properties

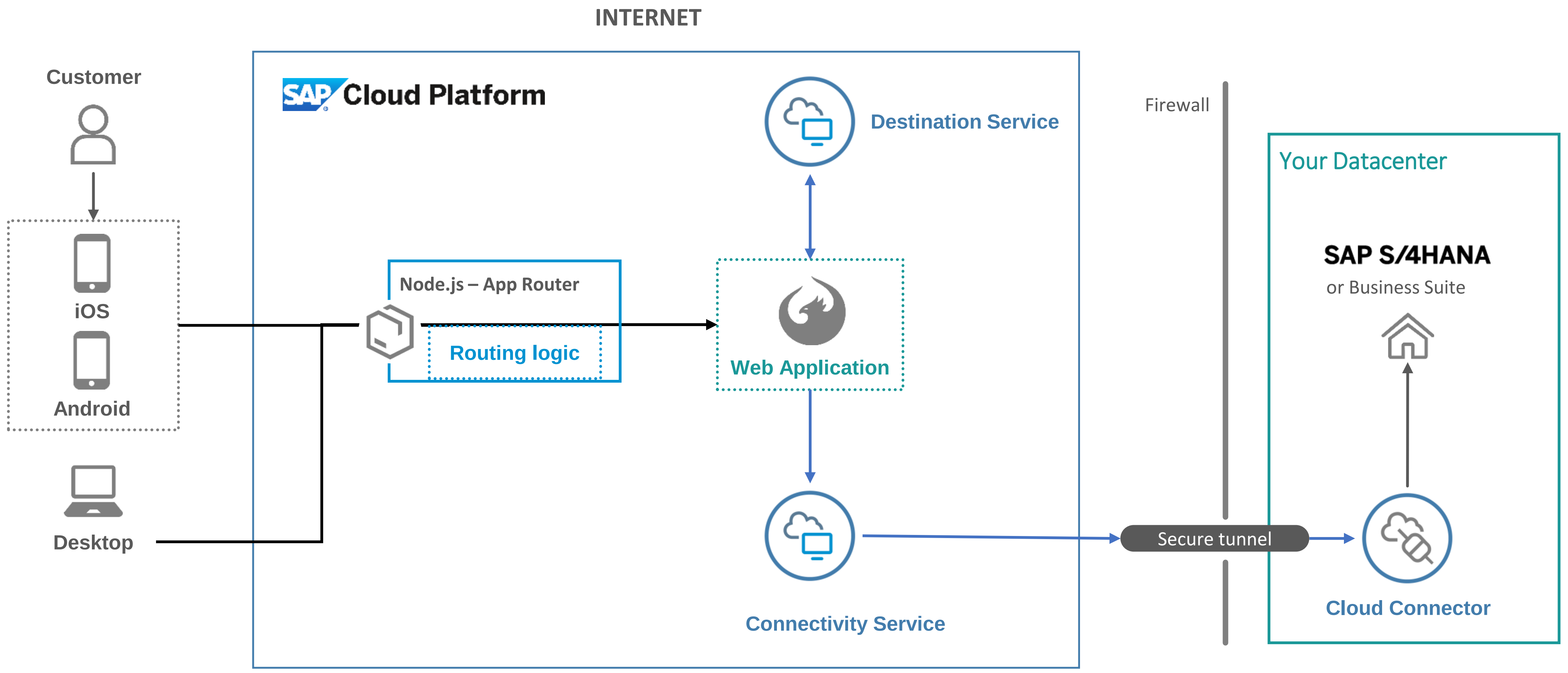
HTML5.Dyna...	true
sap-client	100
WebIDEEnab...	true
WebIDESyst...	sales_api
WebIDEUsage	odata_gen

Connectivity Service – CF

Set up On-Premise communication via HTTP or RFC for your cloud application



```
{
    "tenantmode": "c[redacted]
    "clientid": "sb-clo[redacted]
    "token_service_d[redacted]
    "token_service_u[redacted]
    "xsappname": "c[redacted]
    "onpremise_prox[redacted]
    "onpremise_sock[redacted]
    "clientsecret": "X[redacted]
    "onpremise_prox[redacted]
    "url": "https://ic-c[redacted]
    "onpremise_prox[redacted]
    "uaadomain": "au[redacted]
    "onpremise_prox[redacted]
    "verificationkey": [redacted]
MIICIjANBgkqhkiG9w[redacted]GKMnw7cvCwN
d8rKfYd6olGWigFd+3[redacted]lhKIC7WLwCEJ
lWTxe+FyNklvyZvoLrZ[redacted]JwhTN1HvyXrs
MfeVf0P2th5C9MggY[redacted]) PUBLIC KEY--
    "identityzone": "i[redacted]
    "tenantid": "5349[redacted]
    "onpremise_prox[redacted]
}
```

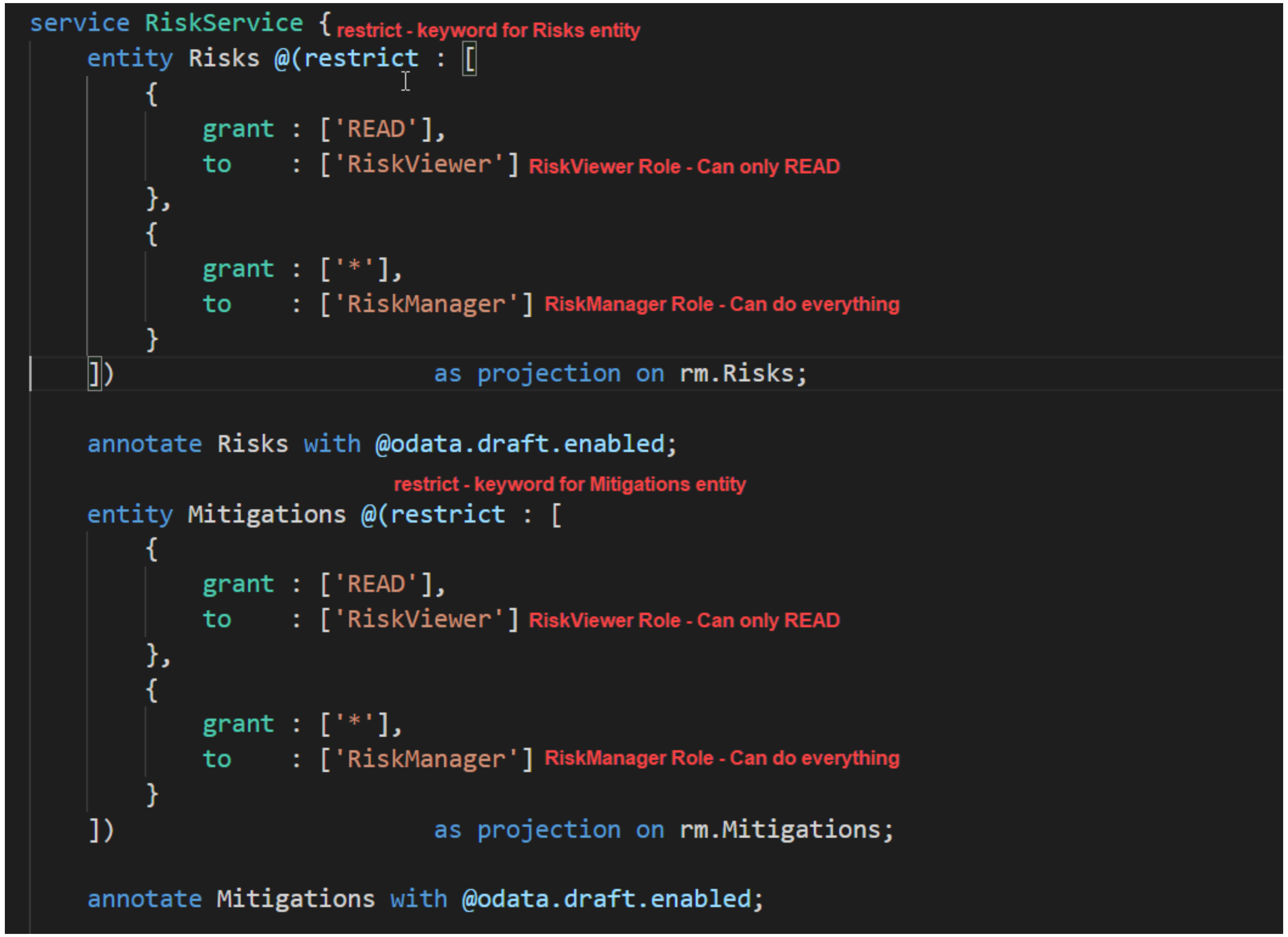

INTERNET
Customer
iOS
Android
Desktop
Cloud Platform
Node.js – App Router
Routing logic
Destination Service
Web Application
Connectivity Service
Firewall
Secure tunnel
Your Datacenter
SAP S/4HANA
or Business Suite
Cloud Connector

Unit 5 – Exercise (External service)

```
git checkout unit5_externalservice (Use tab for branch name)
```

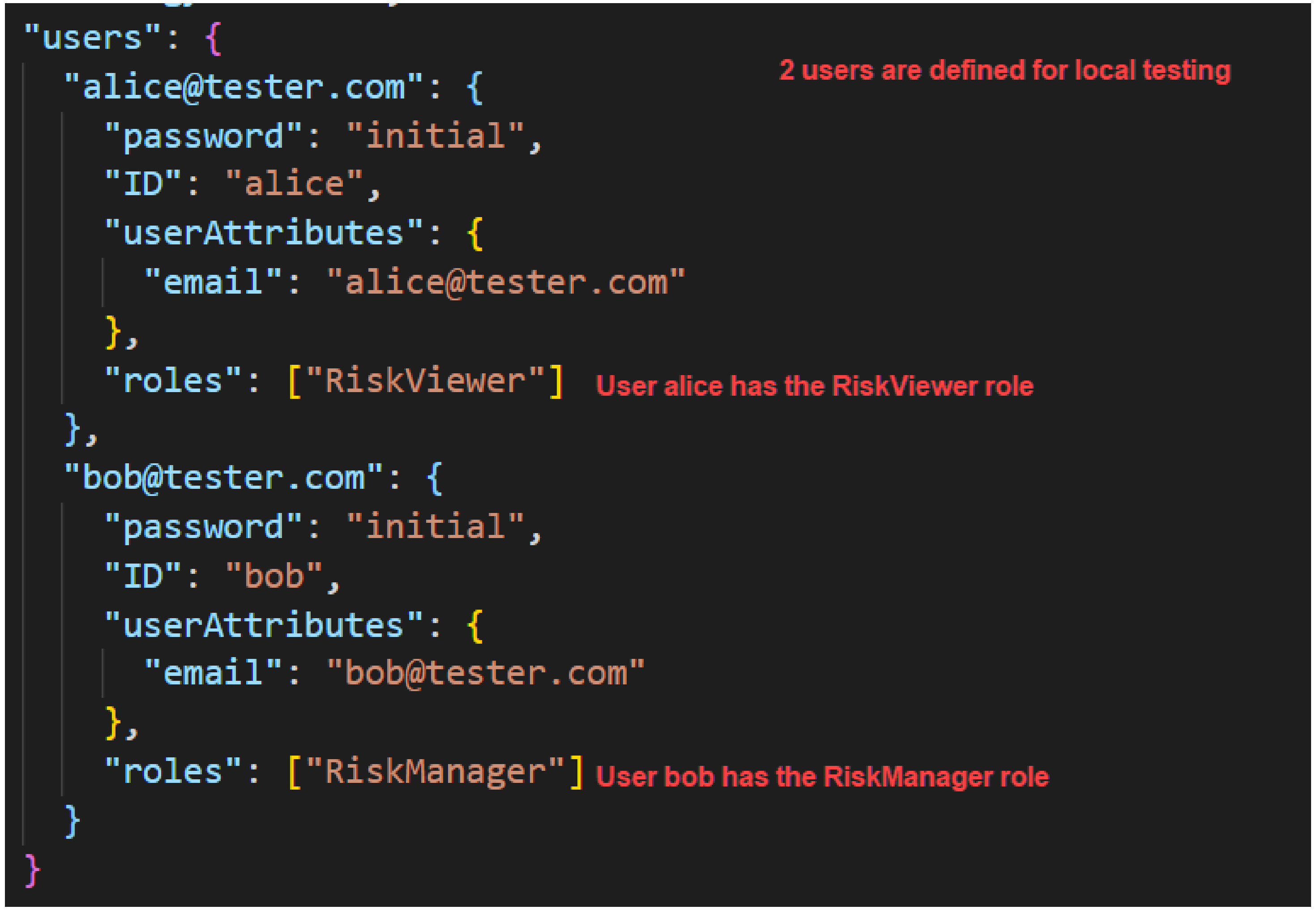
```
Extend CAP service with consumption of external Business Partner
service
```

Unit 6 – Restrictions and Roles



```
service RiskService { restrict - keyword for Risks entity
    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;
                                restrict - keyword for Mitigations entity
    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;
```

Unit 6 – Restrictions and Roles



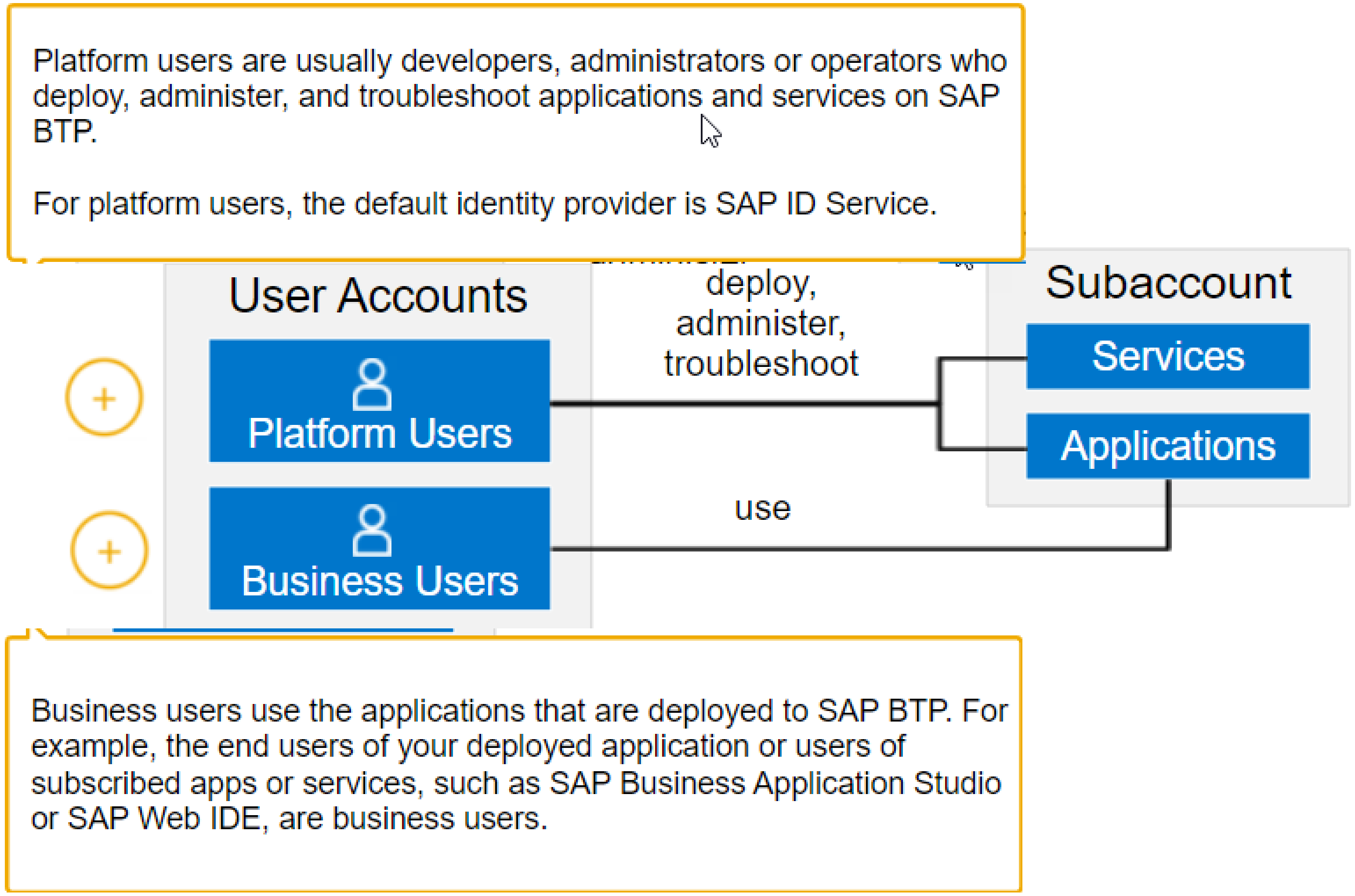
```
"users": {
  "alice@tester.com": {
    "password": "initial",
    "ID": "alice",
    "userAttributes": {
      "email": "alice@tester.com"
    },
    "roles": ["RiskViewer"]
  },
  "bob@tester.com": {
    "password": "initial",
    "ID": "bob",
    "userAttributes": {
      "email": "bob@tester.com"
    },
    "roles": ["RiskManager"]
  }
}
```

2 users are defined for local testing

User alice has the RiskViewer role

User bob has the RiskManager role

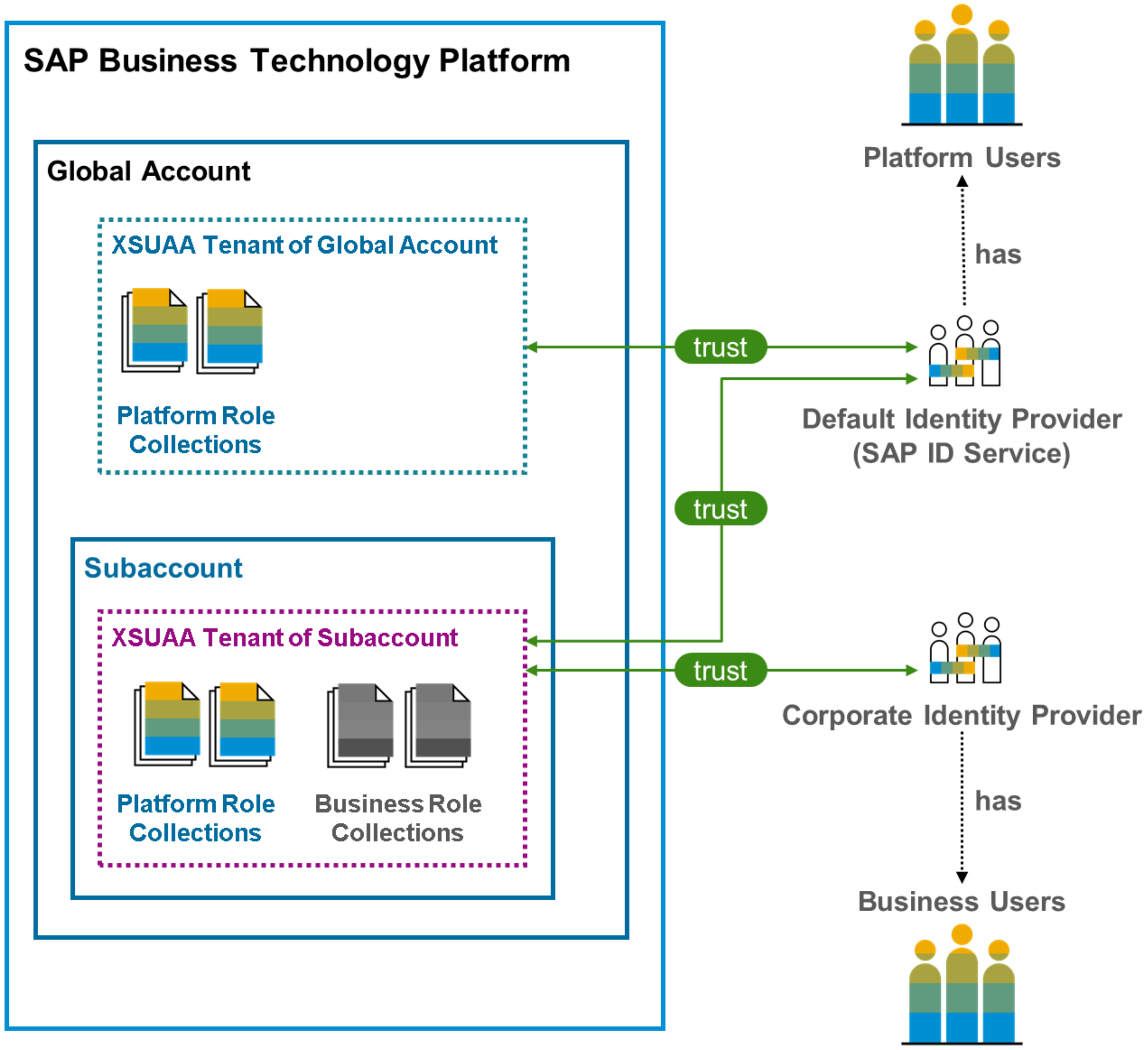
Authorization and Trust Management



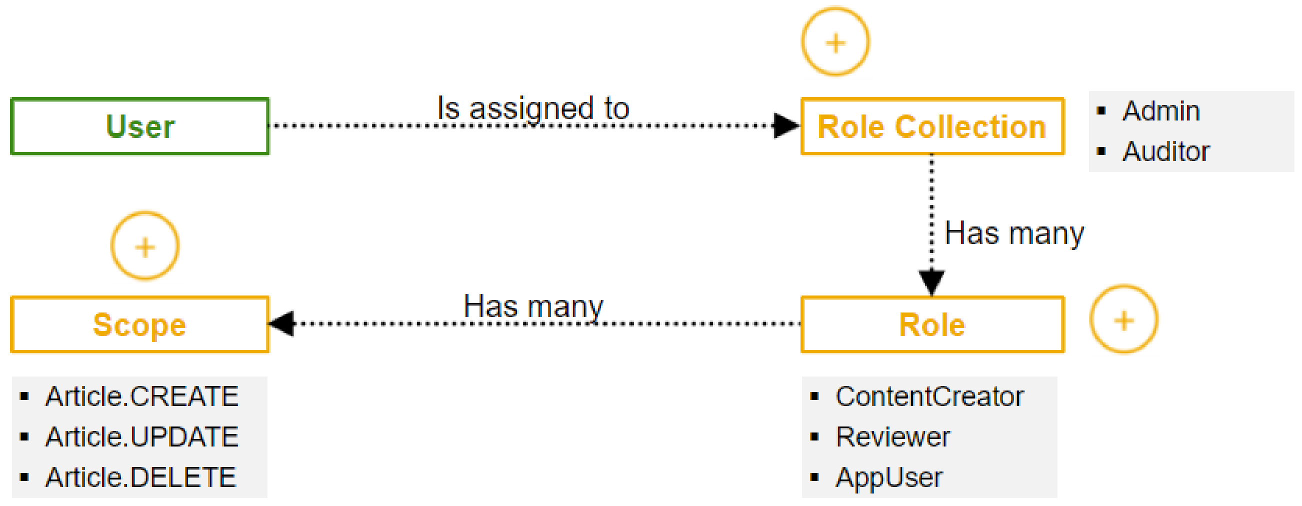
Platform users are usually developers, administrators or operators who deploy, administer, and troubleshoot applications and services on SAP BTP.

For platform users, the default identity provider is SAP ID Service.

Business users use the applications that are deployed to SAP BTP. For example, the end users of your deployed application or users of subscribed apps or services, such as SAP Business Application Studio or SAP Web IDE, are business users.


SAP Business Technology Platform
Global Account
XSUAA Tenant of Global Account
Platform Role Collections
Subaccount
XSUAA Tenant of Subaccount
Platform Role Collections
Business Role Collections
trust
trust
trust
Platform Users
has
Default Identity Provider (SAP ID Service)
Corporate Identity Provider
has
Business Users

Authorization and Trust Management



- 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

Unit 6 – Exercise (Authorization)

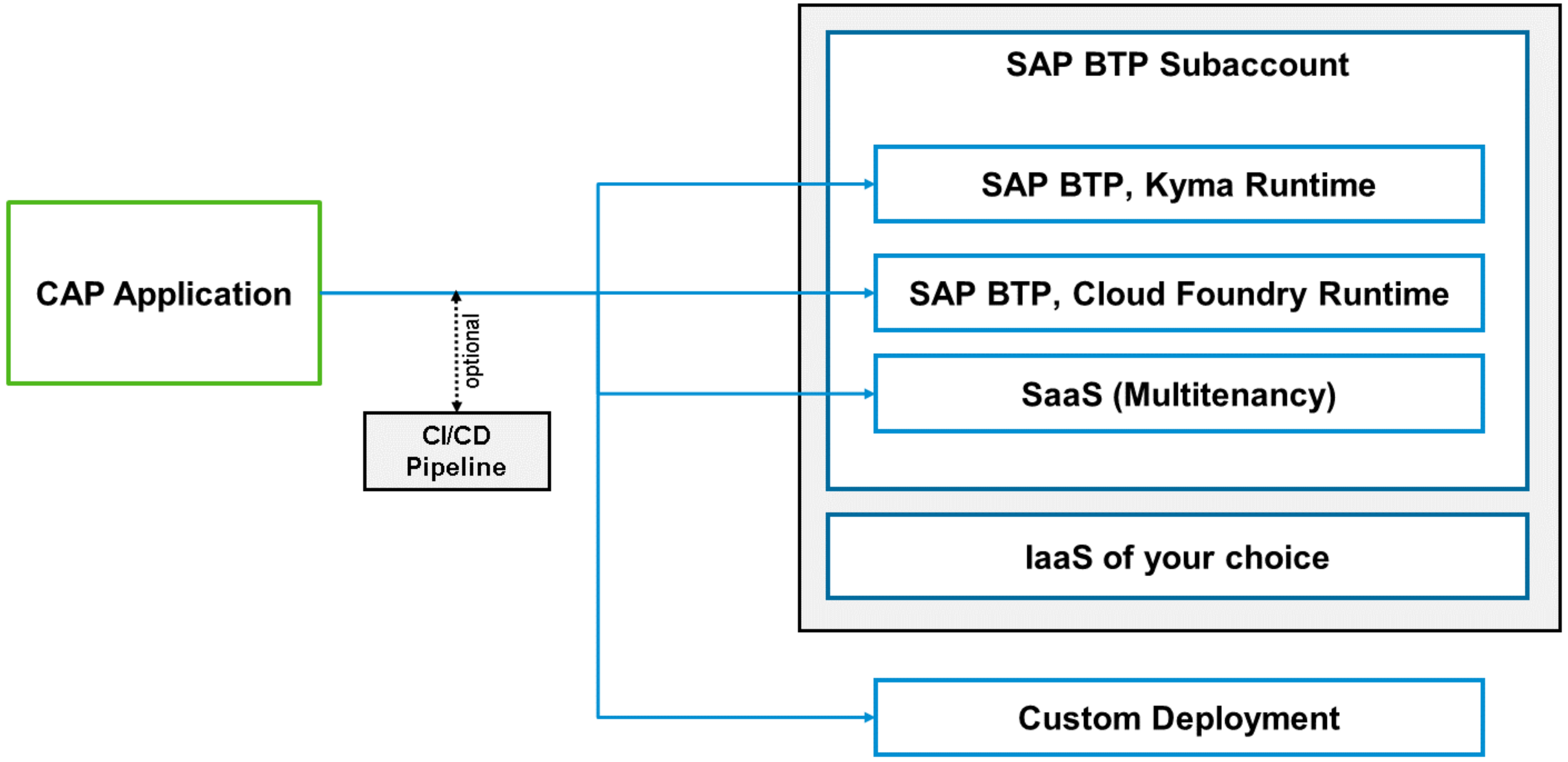
```
git checkout unit6_authorization (Use tab for branch name)
```

Only authorized users can access your app to view and edit data

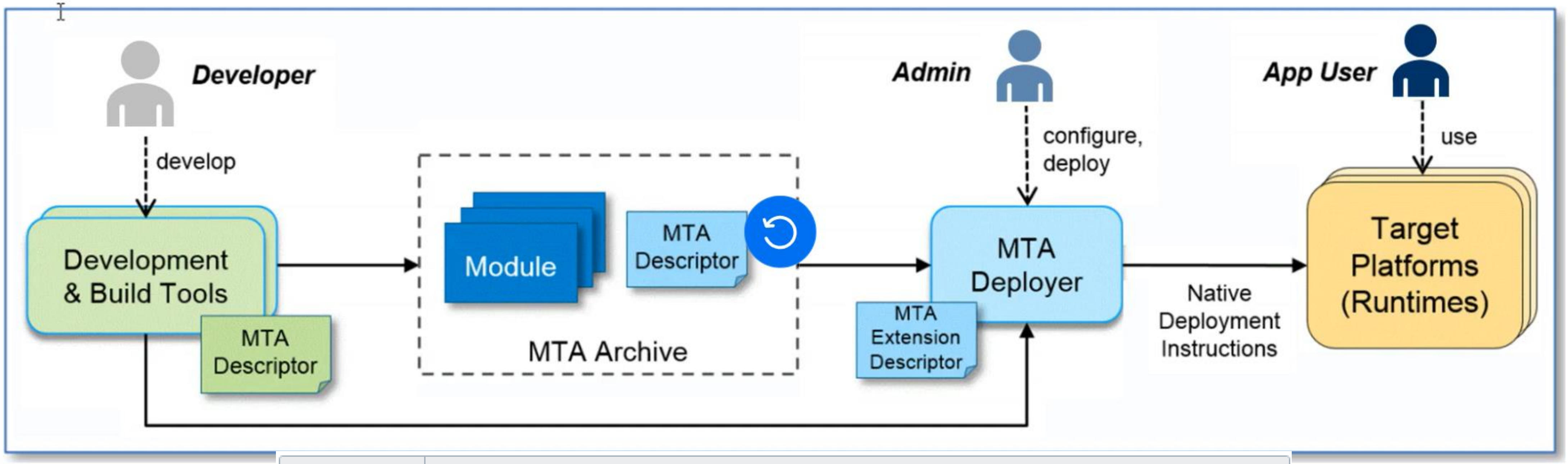
Add authorization to CAP service

Add 2 mock users to further test app locally

Unit 7 – Deployment Options



Unit 7 – MTA Development Descriptor File



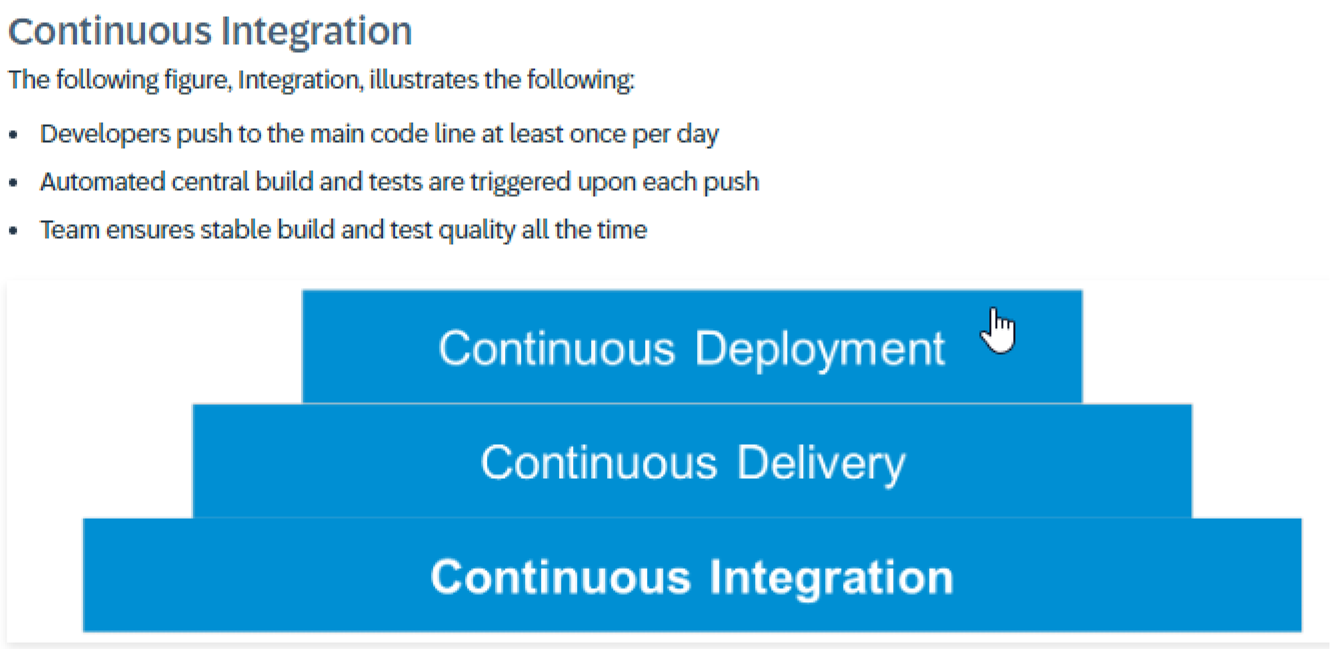
Name	Description
mta.yaml	**Development** descriptor for a multi-target application (MTA). The information in the mta.yaml file provides instructions for the MTA development and build process.
mtad.yaml	**Deployment** descriptor for a multi-target application (MTA). The information in the mad.yaml file provides instructions for the deploy service.
mtaext.yaml	**Deployment extension descriptor** (optional). This is used to provide system-specific details that are not known until deployment time.

Unit 7 – Exercise (Deployment)

```
git checkout unit7_deployment (Use tab for branch name)
```

Deploy CAP application to Cloud Foundry environment of SAP BTP

Unit 8 – Continuous Integration & Delivery



Continuous Integration

The following figure, Integration, illustrates the following:

- Developers push to the main code line at least once per day
- Automated central build and tests are triggered upon each push
- Team ensures stable build and test quality all the time

Deploy an Application

Once you've modeled your application, you can deploy it to use its capabilities.

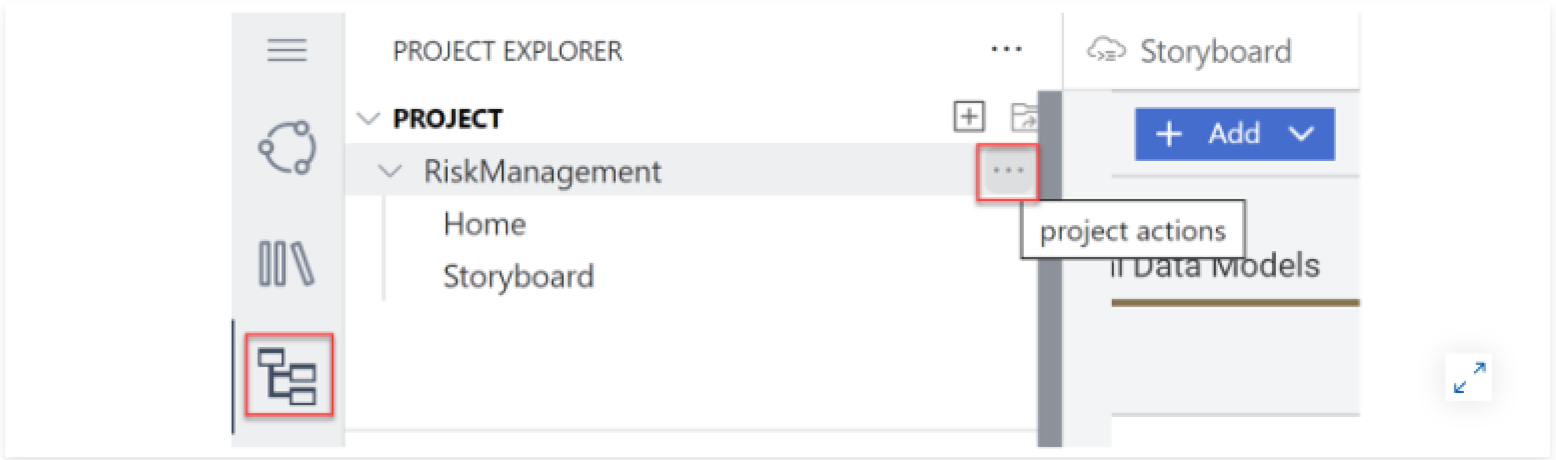
1. From the **Project Explorer**, under **PROJECT**, select ••• **project actions**.
2. Choose **Deploy Project**.
3. You can also view the last deployed project by selecting **View Last Deployed**.
4. Assign the `<applicationname>Manager-<spacename>` role to the users who are allowed to access the deployed application. For details on assigning role collections to users, see Assigning Role Collections to Users or User Groups (you'll be directed to the SAP Business Technology Platform documentation).
5. Once deployment is completed, you can access your subaccount in the SAP BTP cockpit. Navigate to **HTML5 Applications** and choose the required application based on your business needs.
6. In the **Application Overview** page, select the application of your choice to explore it live. Also, you can see the service and metadata details.

Note
You can also deploy your application from the **Task Explorer**. For more information, see Create a Deploy Task.

Sharing and Syncing Applications

You can share and sync your applications with other developers for them to contribute and enhance your applications by connecting to a Git repository. For more information, see Git Source Control.

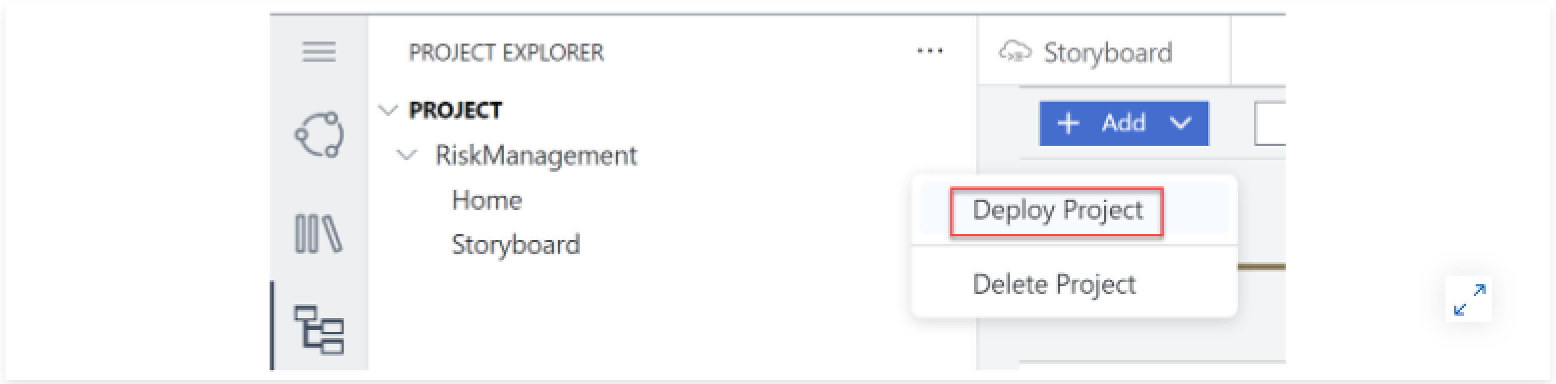
b. Find the three little dots, in the Project Explorer, right to your project's name.



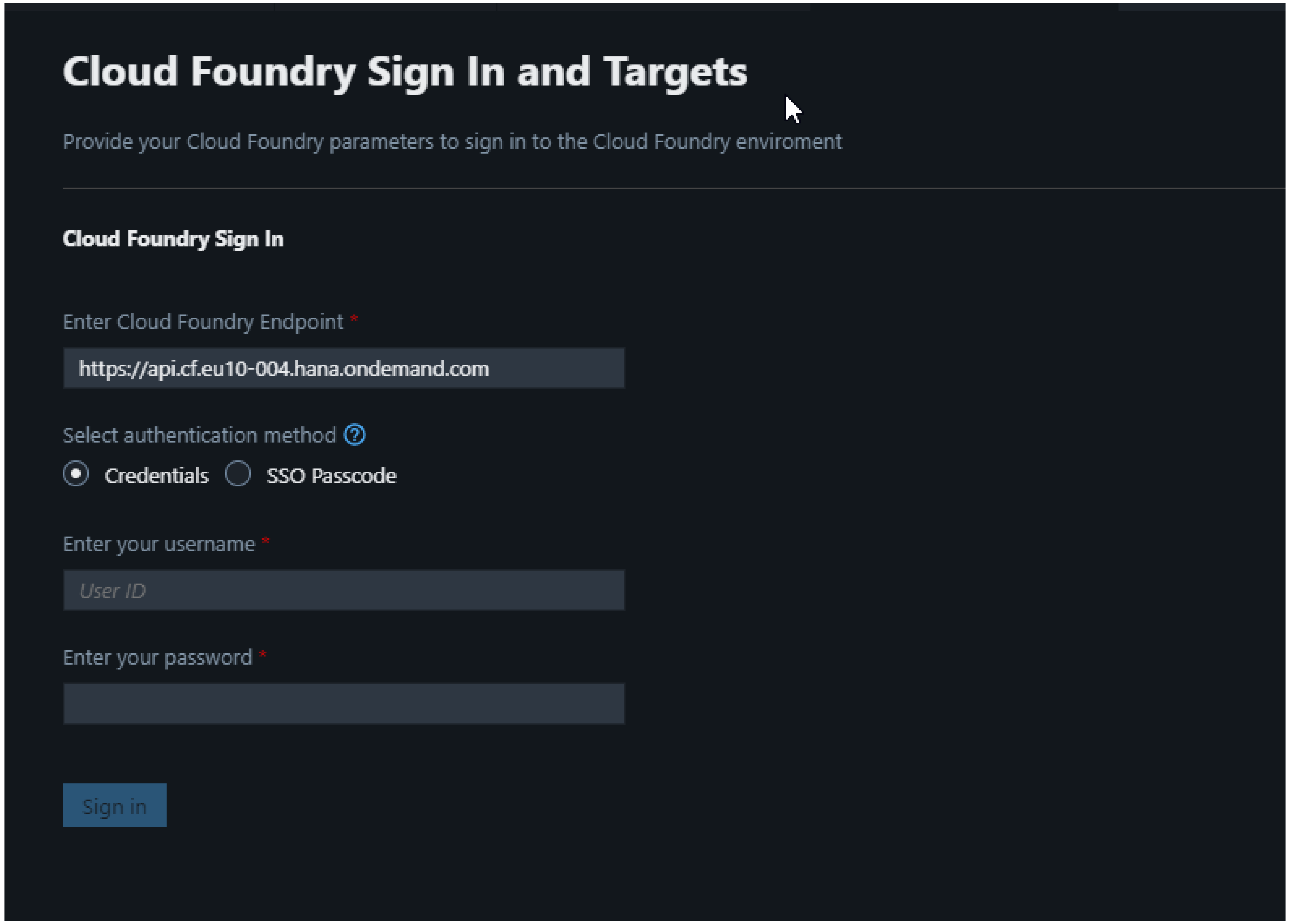
If you select those, a new menu will open.

c. Select them.

d. Select *Deploy Project*



e. Check, that your application deploys.



Cloud Foundry Sign In and Targets

Provide your Cloud Foundry parameters to sign in to the Cloud Foundry enviroment

Cloud Foundry Sign In

Enter Cloud Foundry Endpoint *

https://api.cf.eu10-004.hana.ondemand.com

Select authentication method

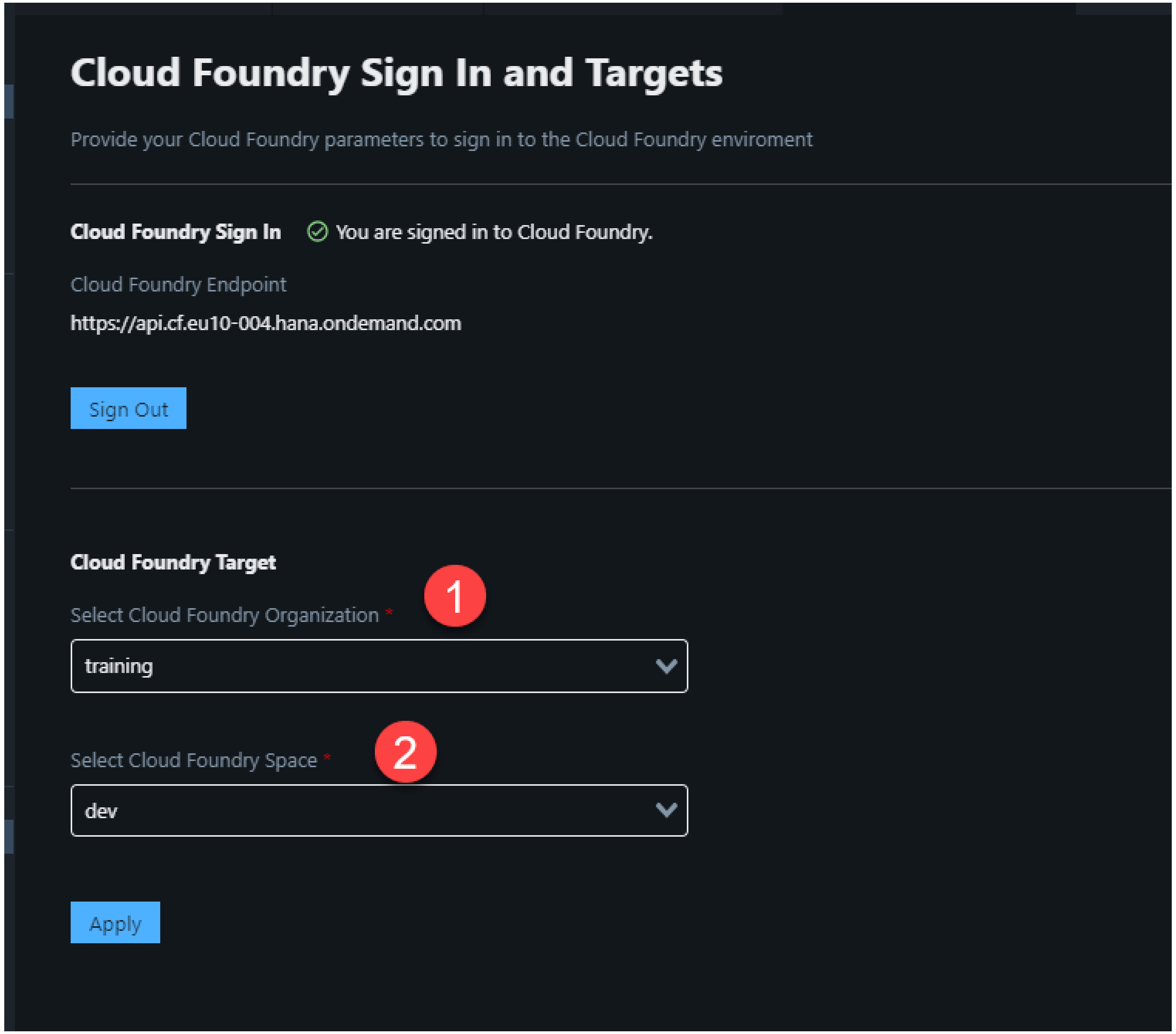
[✓ Credentials] SSO Passcode

Enter your username *

User ID

Enter your password *

Sign in



Cloud Foundry Sign In and Targets

Provide your Cloud Foundry parameters to sign in to the Cloud Foundry enviroment

Cloud Foundry Sign In ✓ You are signed in to Cloud Foundry.

Cloud Foundry Endpoint

https://api.cf.eu10-004.hana.ondemand.com

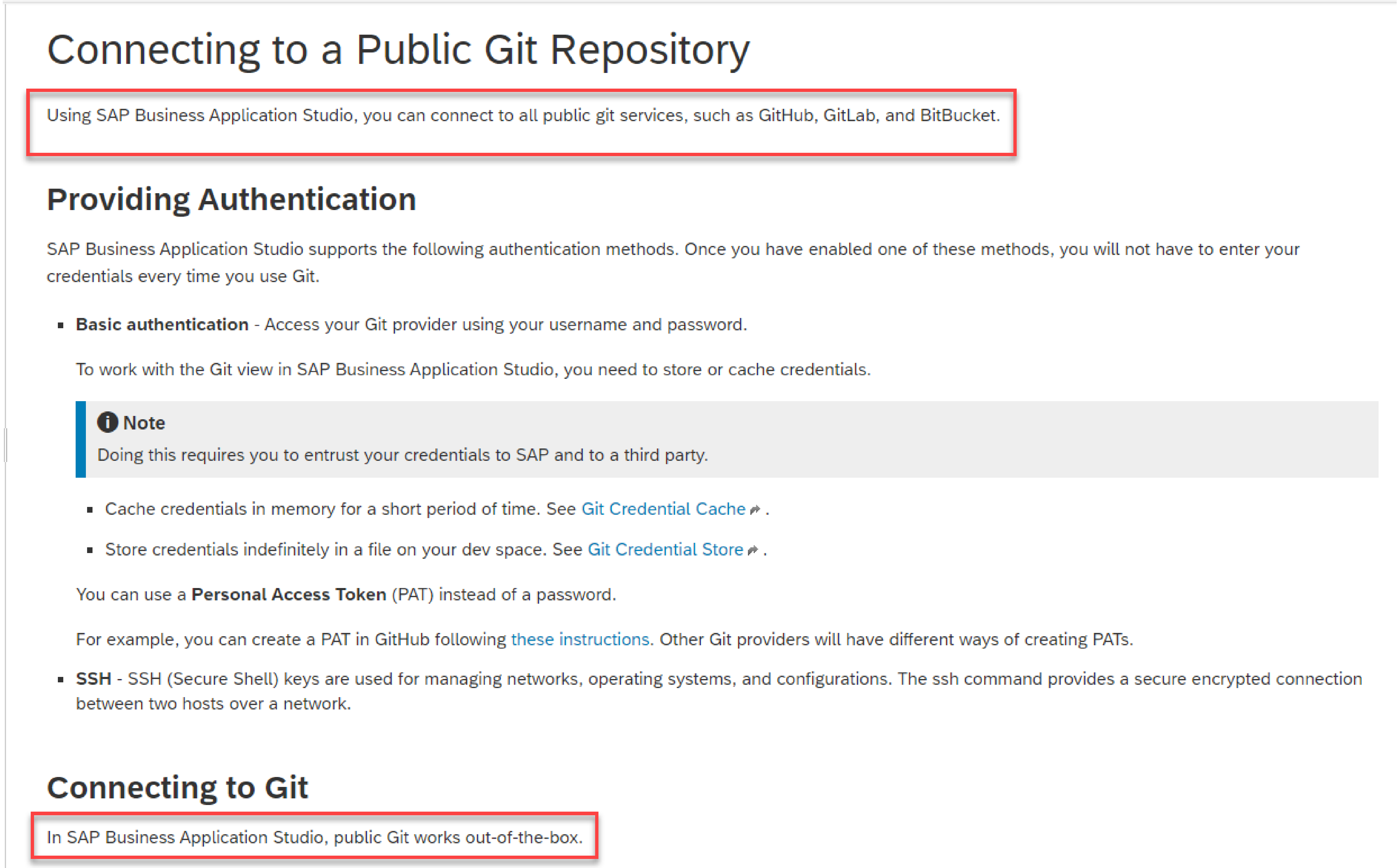
Cloud Foundry Target

Select Cloud Foundry Organization *

training

Select Cloud Foundry Space *

dev



Connecting to a Public Git Repository

Using SAP Business Application Studio, you can connect to all public git services, such as GitHub, GitLab, and BitBucket.

Providing Authentication

SAP Business Application Studio supports the following authentication methods. Once you have enabled one of these methods, you will not have to enter your credentials every time you use Git.

- **Basic authentication** - Access your Git provider using your username and password.

  To work with the Git view in SAP Business Application Studio, you need to store or cache credentials.

  > **Note**
  > Doing this requires you to entrust your credentials to SAP and to a third party.

  - Cache credentials in memory for a short period of time. See Git Credential Cache .
  - Store credentials indefinitely in a file on your dev space. See Git Credential Store .

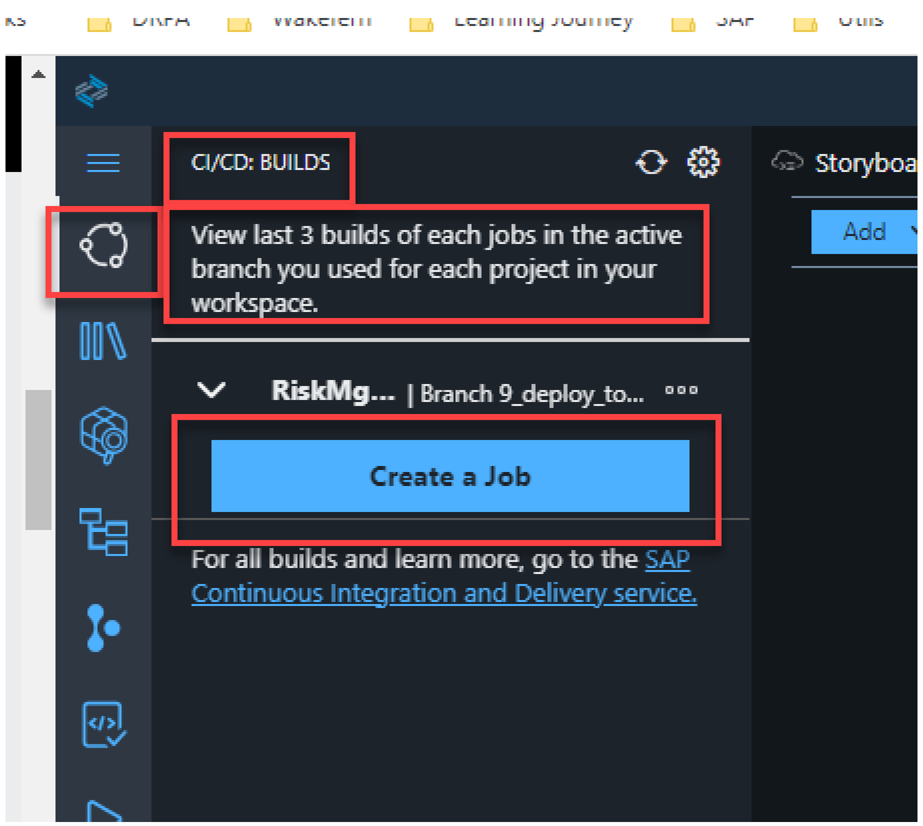
  You can use a **Personal Access Token** (PAT) instead of a password.

  For example, you can create a PAT in GitHub following these instructions. Other Git providers will have different ways of creating PATs.

- **SSH** - SSH (Secure Shell) keys are used for managing networks, operating systems, and configurations. The ssh command provides a secure encrypted connection between two hosts over a network.

Connecting to Git

In SAP Business Application Studio, public Git works out-of-the-box.



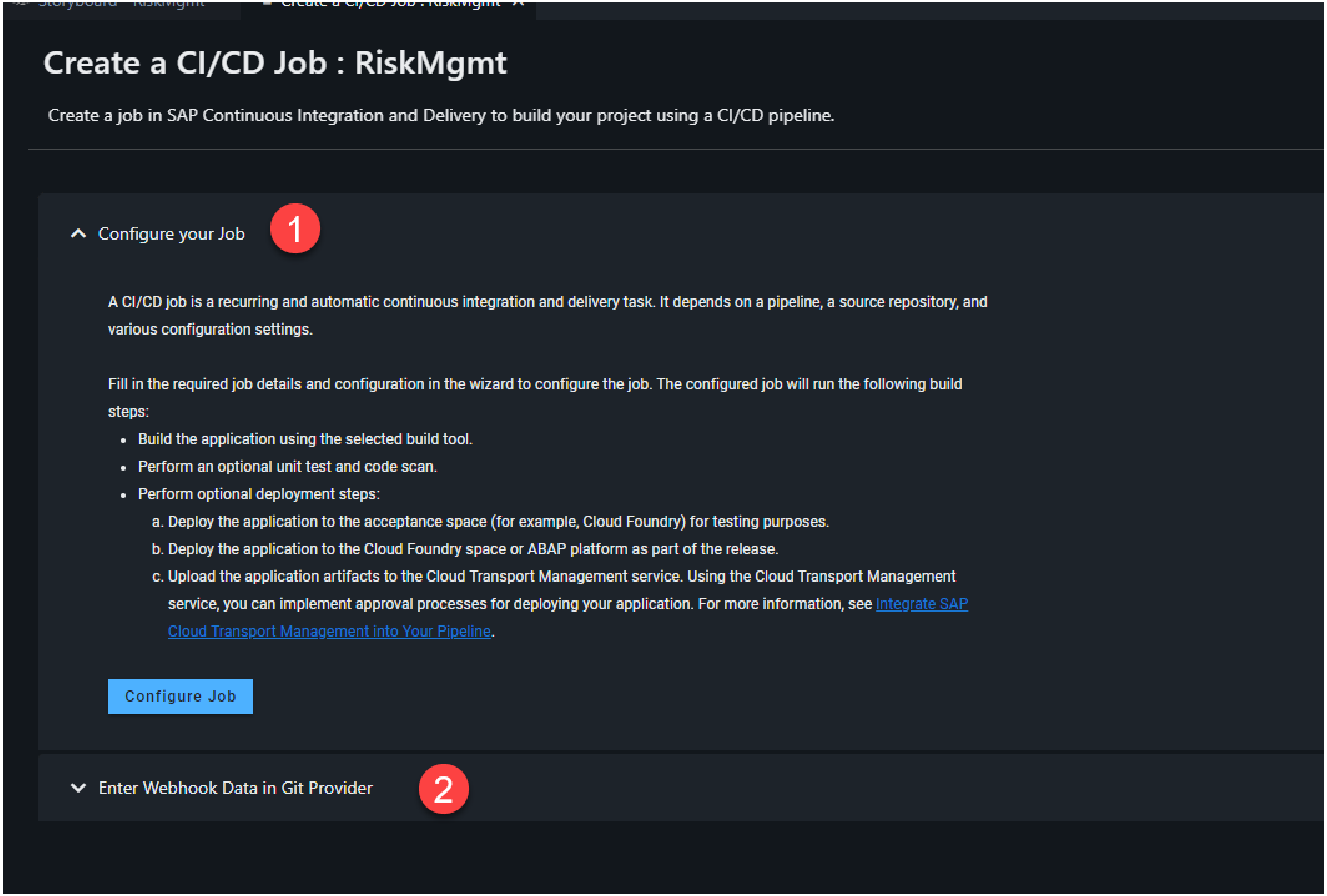
CI/CD: BUILDS

View last 3 builds of each jobs in the active branch you used for each project in your workspace.

RiskMg... | Branch 9_deploy_to...

Create a Job

For all builds and learn more, go to the SAP Continuous Integration and Delivery service.


Create a CI/CD Job : RiskMgmt

Create a job in SAP Continuous Integration and Delivery to build your project using a CI/CD pipeline.

Configure your Job

1

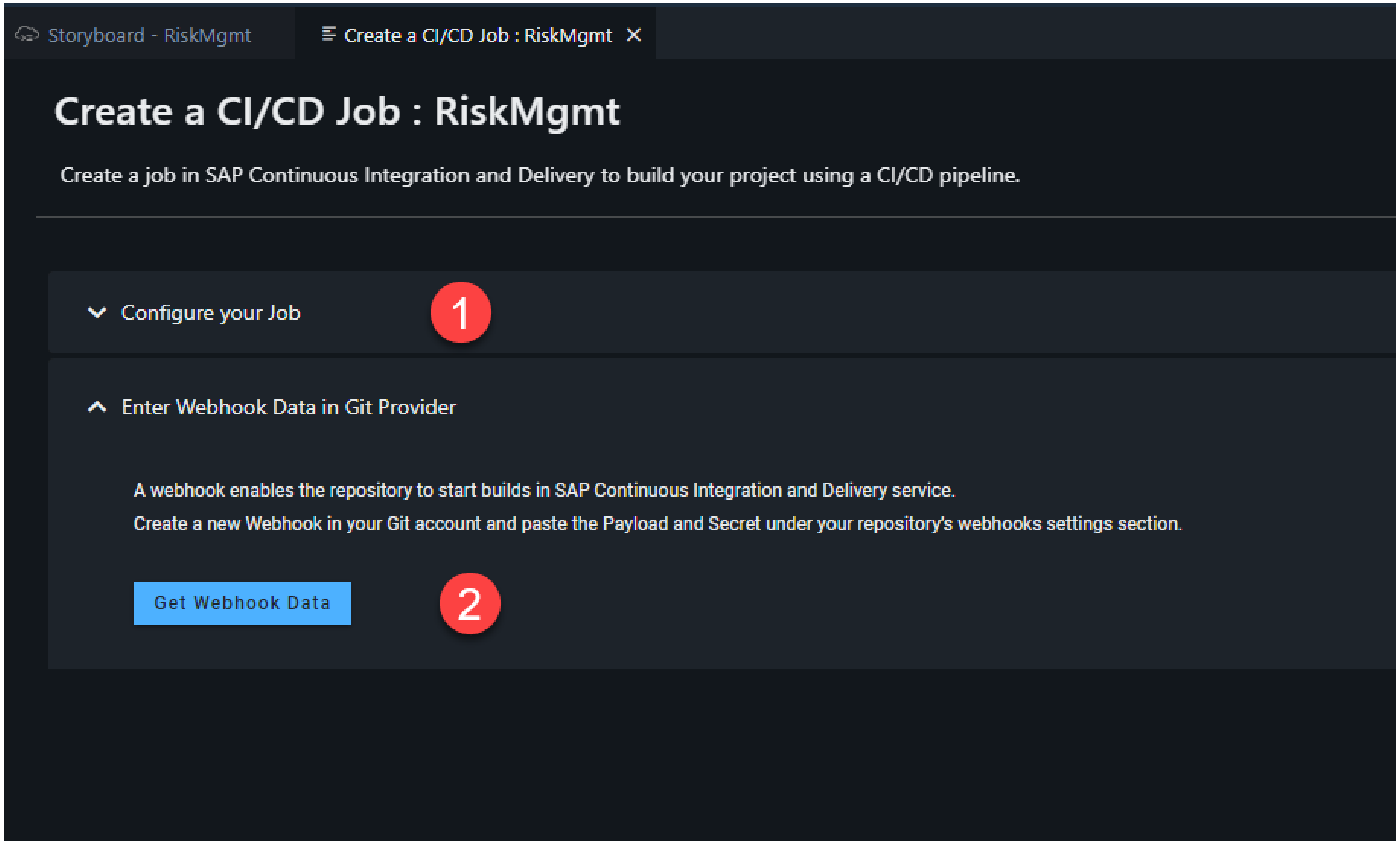
A CI/CD job is a recurring and automatic continuous integration and delivery task. It depends on a pipeline, a source repository, and various configuration settings.

Fill in the required job details and configuration in the wizard to configure the job. The configured job will run the following build steps:

- Build the application using the selected build tool.
- Perform an optional unit test and code scan.
- Perform optional deployment steps:
  a. Deploy the application to the acceptance space (for example, Cloud Foundry) for testing purposes.
  b. Deploy the application to the Cloud Foundry space or ABAP platform as part of the release.
  c. Upload the application artifacts to the Cloud Transport Management service. Using the Cloud Transport Management service, you can implement approval processes for deploying your application. For more information, see Integrate SAP Cloud Transport Management into Your Pipeline.

Enter Webhook Data in Git Provider

2

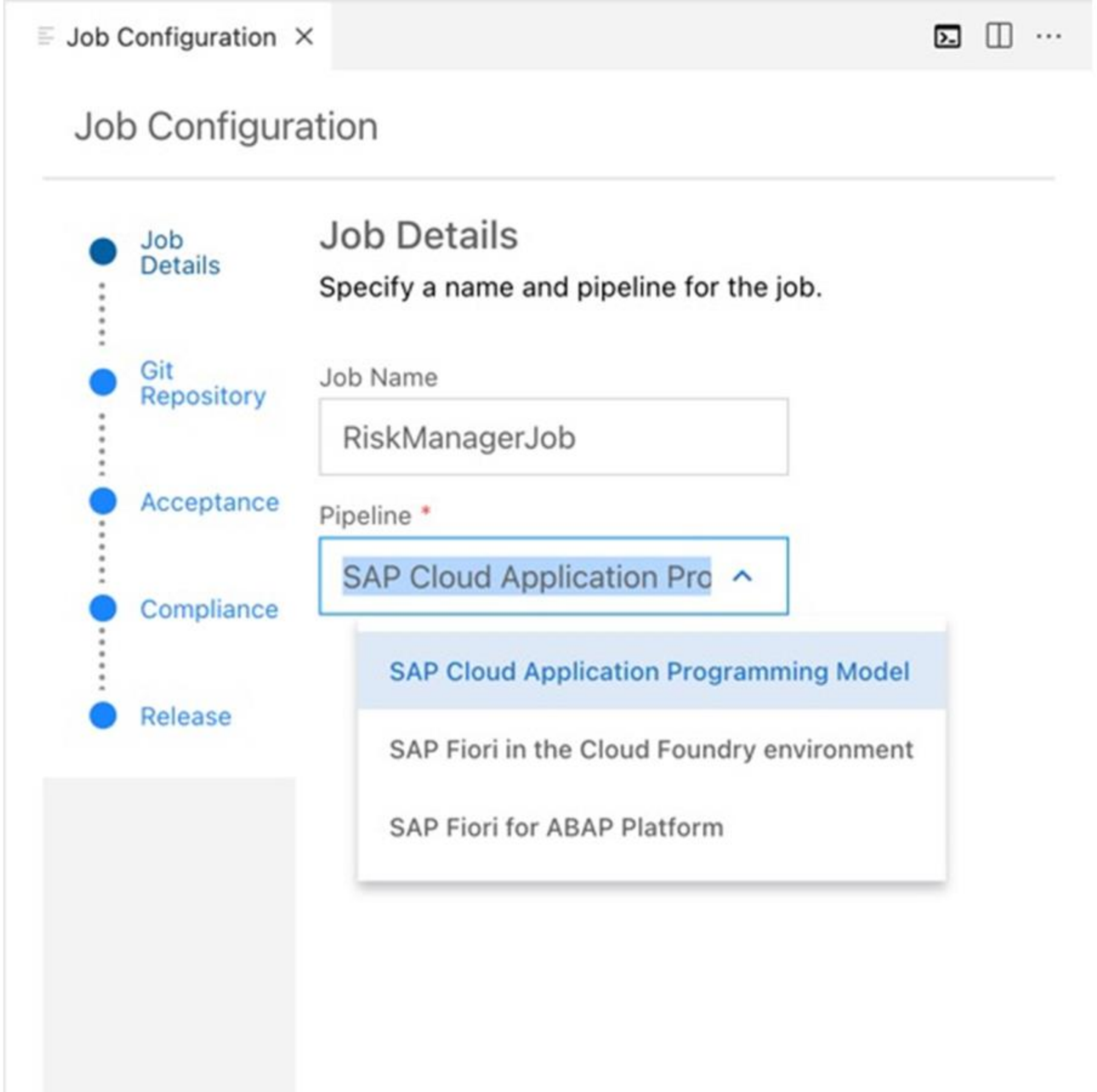

Storyboard - RiskMgmt
Create a CI/CD Job : RiskMgmt

Create a CI/CD Job : RiskMgmt

Create a job in SAP Continuous Integration and Delivery to build your project using a CI/CD pipeline.

Configure your Job
1
Enter Webhook Data in Git Provider

A webhook enables the repository to start builds in SAP Continuous Integration and Delivery service.
Create a new Webhook in your Git account and paste the Payload and Secret under your repository's webhooks settings section.



Job Configuration

Job Configuration

Job Details

Git Repository

Acceptance

Compliance

Release

Job Details

Specify a name and pipeline for the job.

Job Name

RiskManagerJob

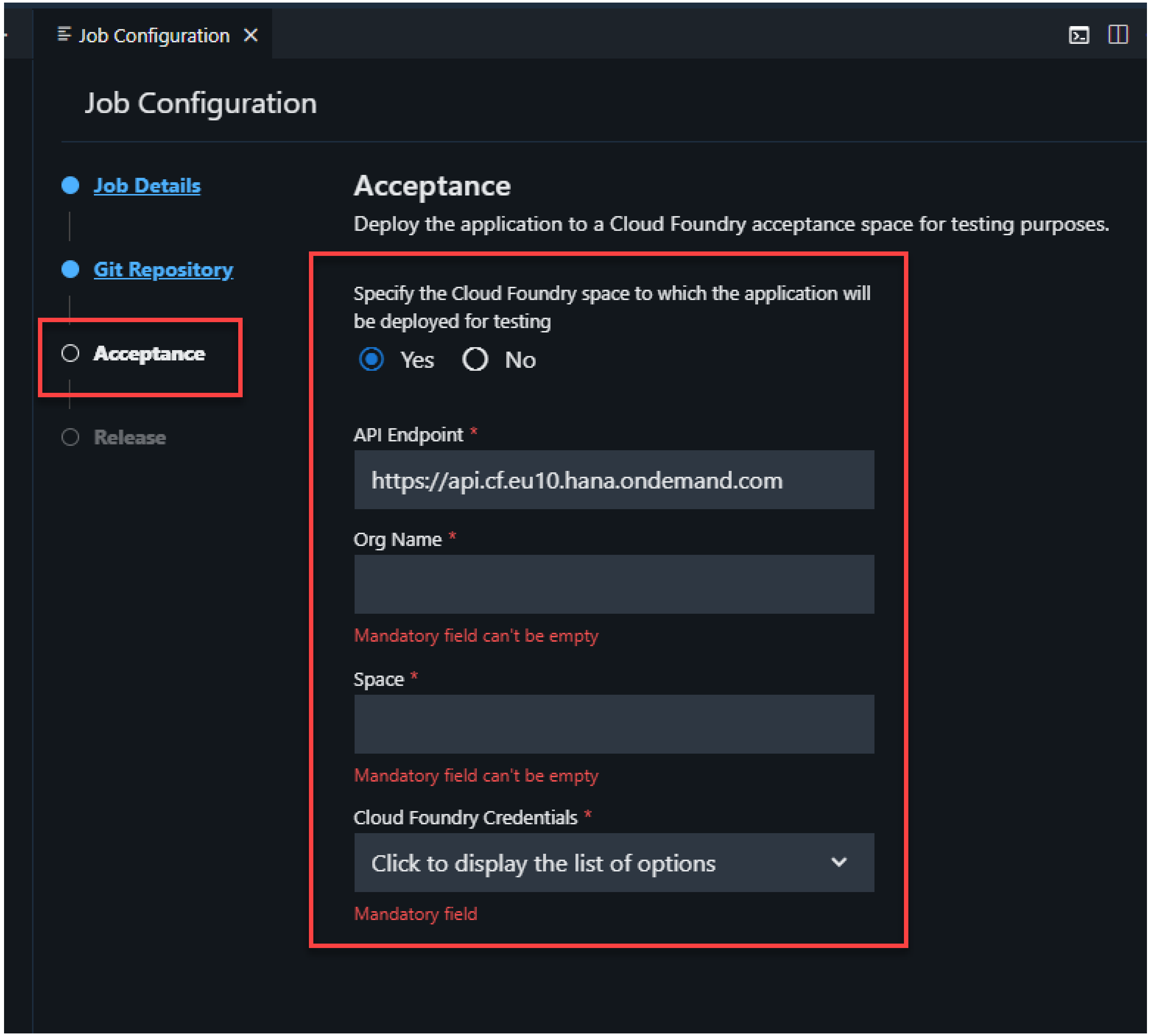
Pipeline *

SAP Cloud Application Pro

SAP Cloud Application Programming Model

SAP Fiori in the Cloud Foundry environment

SAP Fiori for ABAP Platform



Job Configuration

- ● Job Details
- ● Git Repository
- ○ Acceptance
- ○ Release

Acceptance

Deploy the application to a Cloud Foundry acceptance space for testing purposes.

Specify the Cloud Foundry space to which the application will be deployed for testing

[✓ Yes] No

API Endpoint *

https://api.cf.eu10.hana.ondemand.com

Org Name *

Mandatory field can't be empty

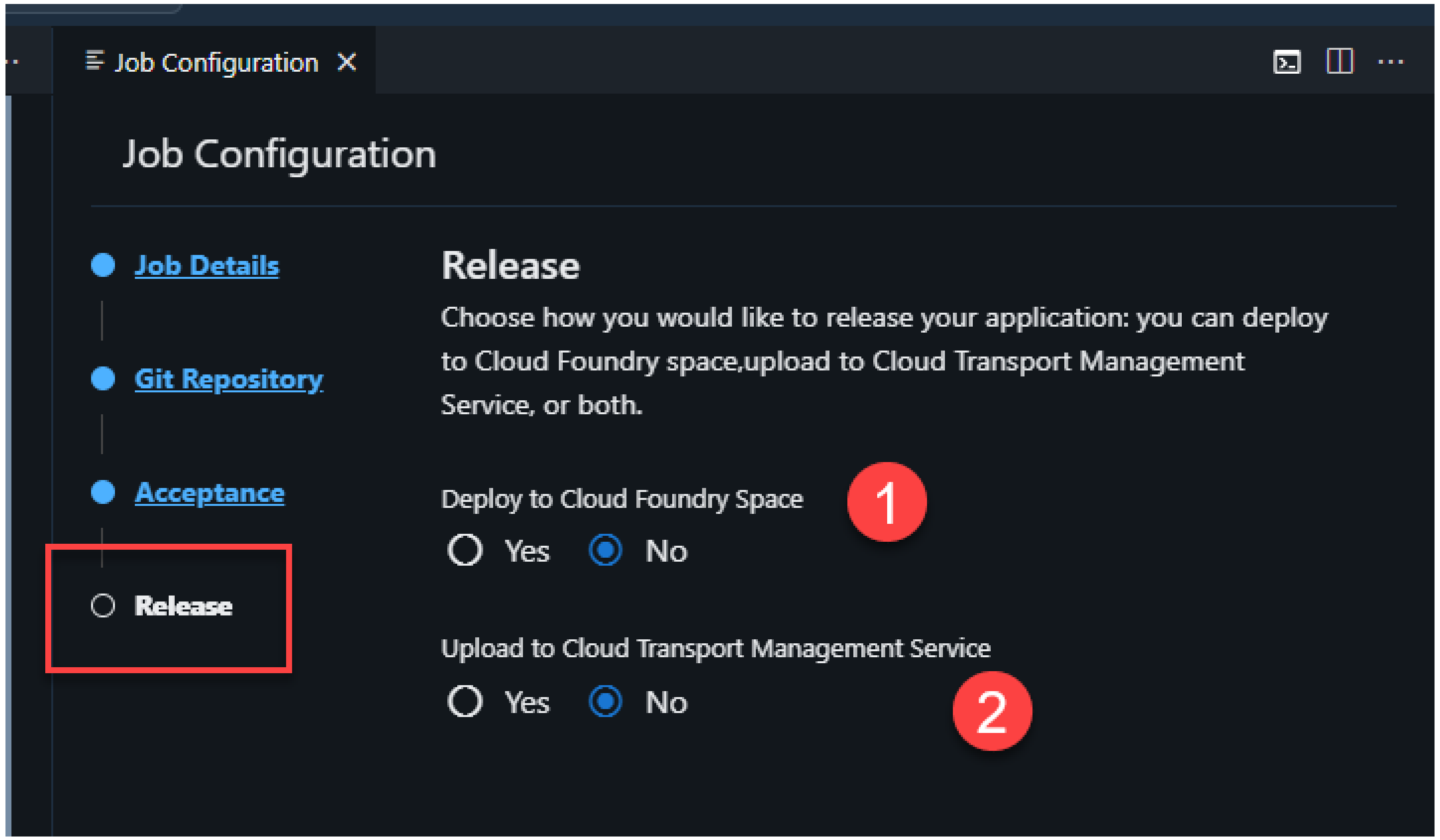
Space *

Mandatory field can't be empty

Cloud Foundry Credentials *

Click to display the list of options

Mandatory field



Job Configuration

Job Configuration

Release

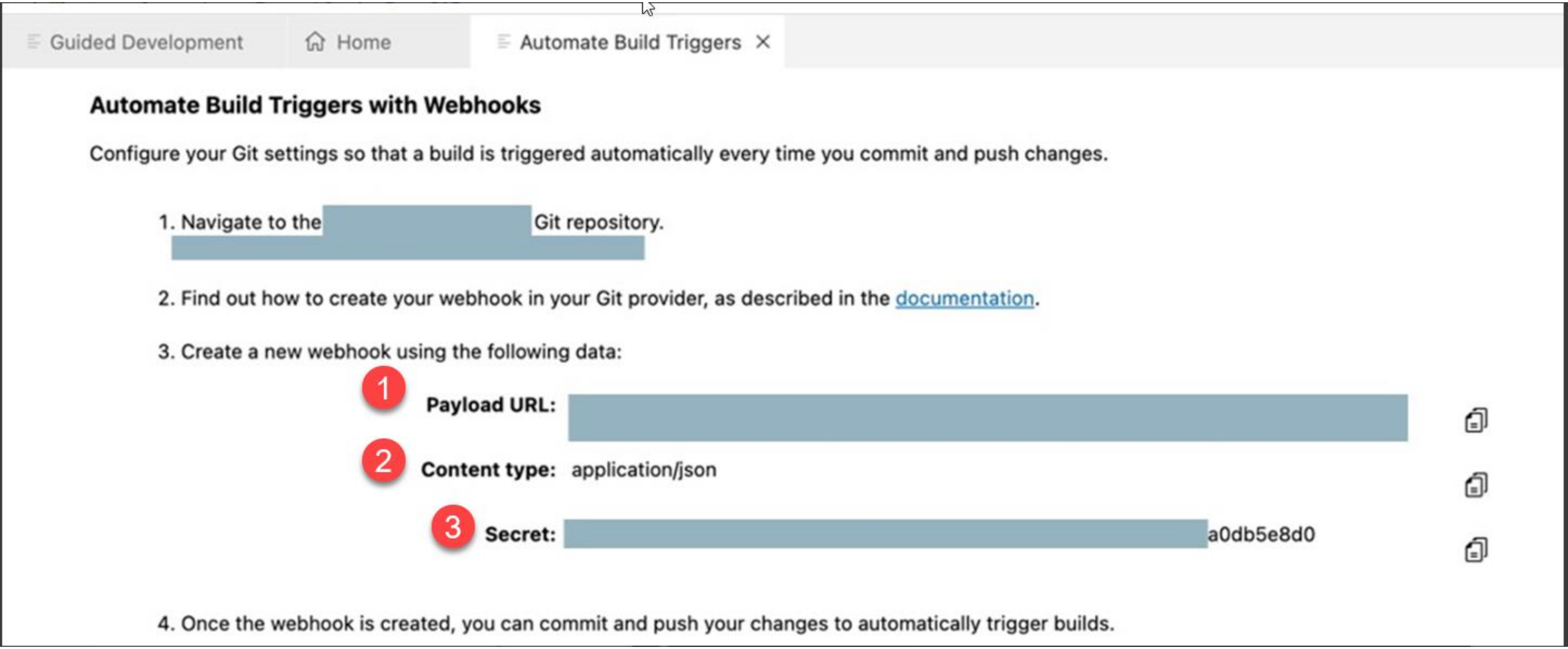
Choose how you would like to release your application: you can deploy to Cloud Foundry space,upload to Cloud Transport Management Service, or both.

Deploy to Cloud Foundry Space

Yes [✓ No]

Upload to Cloud Transport Management Service

Yes [✓ No]



Automate Build Triggers with Webhooks

Configure your Git settings so that a build is triggered automatically every time you commit and push changes.

1. Navigate to the [redacted] Git repository.
[redacted]

2. Find out how to create your webhook in your Git provider, as described in the documentation.

3. Create a new webhook using the following data:

Payload URL: [redacted]

Content type: application/json

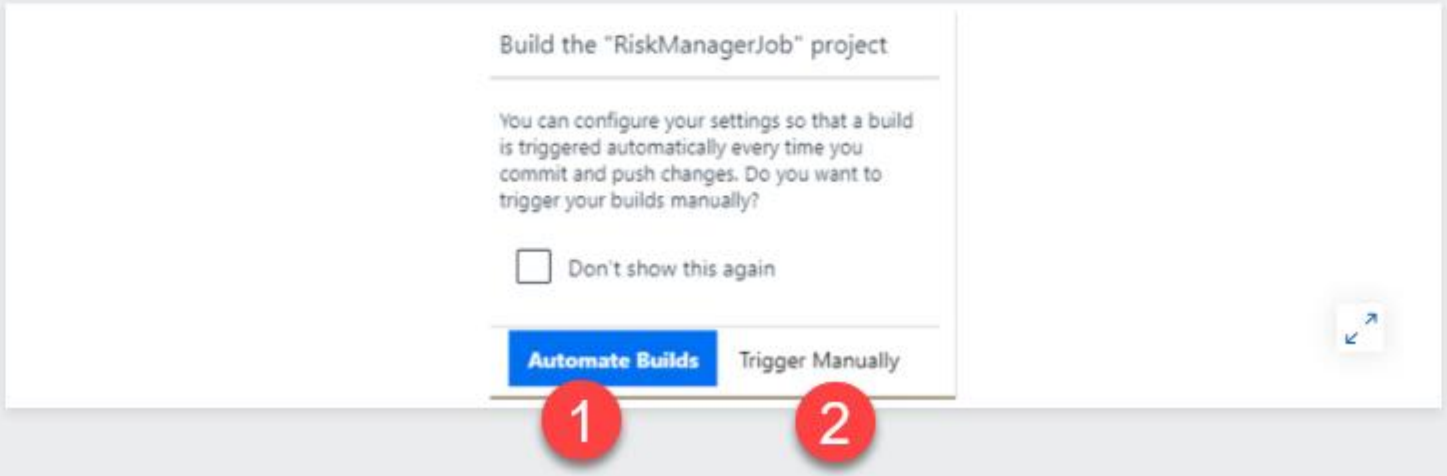
Secret: [redacted]a0db5e8d0

4. Once the webhook is created, you can commit and push your changes to automatically trigger builds.

Note

In case the build is not triggered automatically, please do the following:

1. Choose the Overflow button beside the job name to open the context menu.
2. Choose *Trigger a build*.
3. In the confirmation dialog, you can choose whether you want your builds to be triggered automatically or manually in the future upon committing changes to your Git repository.



Build the "RiskManagerJob" project

You can configure your settings so that a build is triggered automatically every time you commit and push changes. Do you want to trigger your builds manually?

☐ Don't show this again