Unit 1, 2 – Developing with SAP Extension Suite

Certification: C_CPE_13

Unit 1 – SAP Business Technology Platform (BTP)

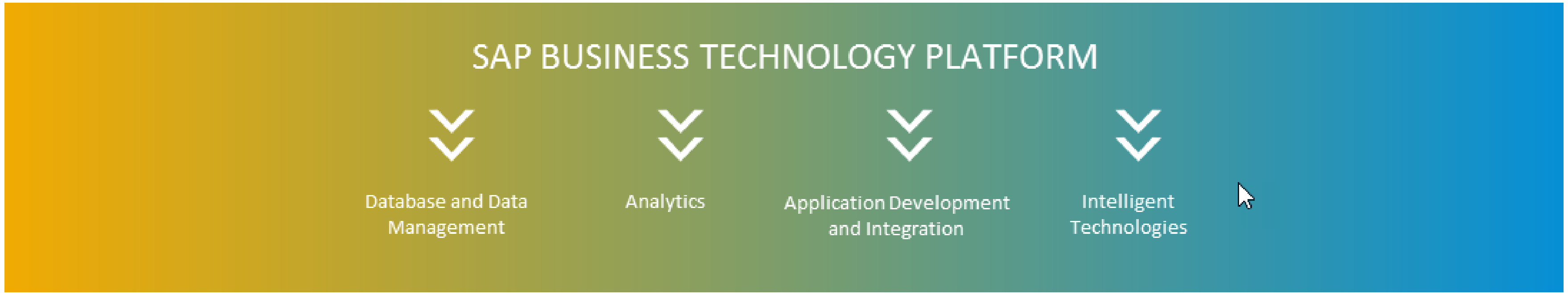
Unit 2 – Development Environment and CAP-Project

Agenda

- Introduction to SAP BTP
- Introduction to SAP Business Application Studio
- App Demo
- Initialize empty project
- Create data model
- Create OData Service
- Summary Points

Open runtime
Open API
Open environment

Foundational services



SAP BUSINESS TECHNOLOGY PLATFORM

Database and Data Management
Analytics
Application Development and Integration
Intelligent Technologies

Cloud,
on premise,
hybrid, edge

Any data store connectivity

- SAP HANA
- SAP HANA Cloud
- SAP Data Intelligence
- SAP Master Data Governance
- SAP Adaptive Server Enterprise
- ... *

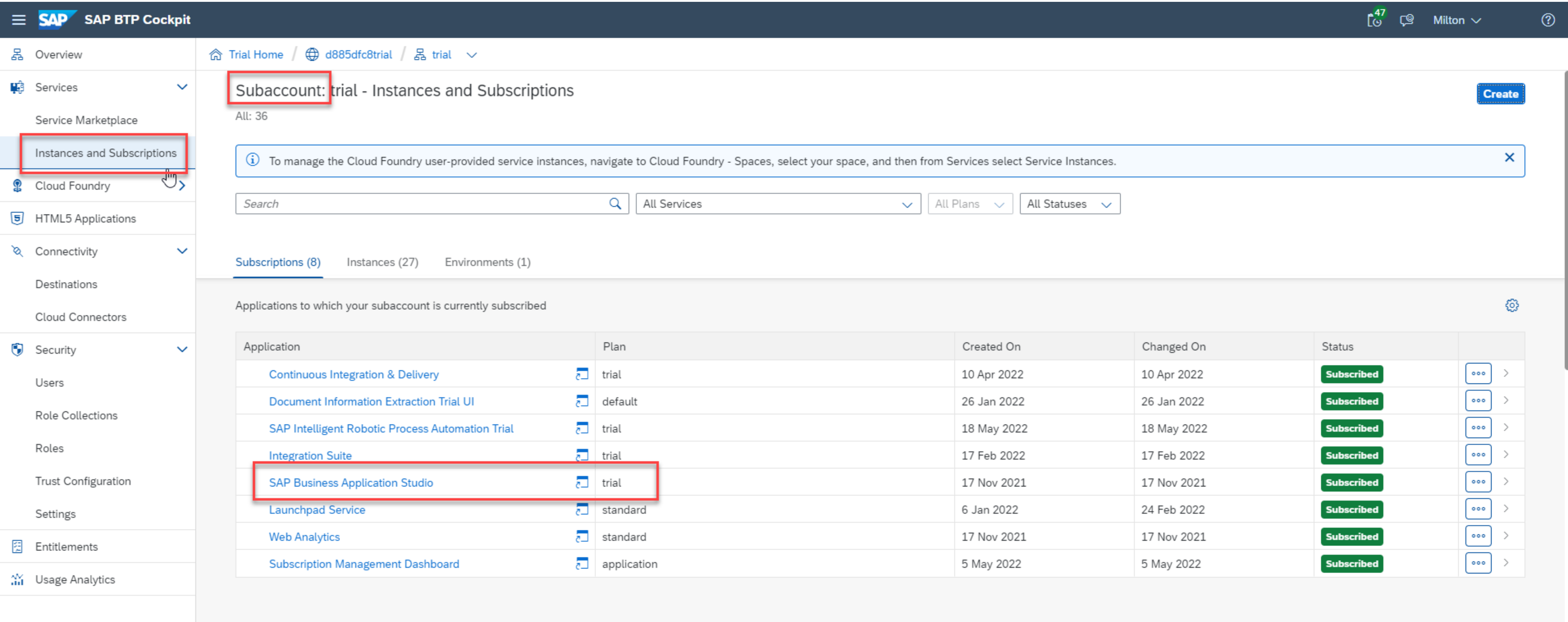
- SAP Analytics Cloud
- SAP Data Warehouse Cloud
- SAP BusinessObjects BI
- SAP Business Planning and Consolidation
- SAP BW/4HANA
- ... *

- SAP Integration Suite
- SAP Extension Suite
- SAP Process Orchestration
- ... *

- SAP Intelligent Robotic Process Automation
- SAP Conversational AI
- SAP Internet of Things
- SAP AI Business Services
- SAP Ruum
- ... *

* Representative list: not exhaustive nor inclusive of all offerings

SAP Business Application Studio



Subaccount: trial - Instances and Subscriptions

All: 36

To manage the Cloud Foundry user-provided service instances, navigate to Cloud Foundry - Spaces, select your space, and then from Services select Service Instances.

Applications to which your subaccount is currently subscribed

Application	Plan	Created On	Changed On	Status
Continuous Integration & Delivery	trial	10 Apr 2022	10 Apr 2022	Subscribed
Document Information Extraction Trial UI	default	26 Jan 2022	26 Jan 2022	Subscribed
SAP Intelligent Robotic Process Automation Trial	trial	18 May 2022	18 May 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	5 May 2022	5 May 2022	Subscribed

Application Demo

Application Features

- OData V4 Service
- SAP Fiori Elements Application
- External Services from SAP S/4HANA Cloud
- Manual and Automated Deployment
- Security – Authentication and Authorization

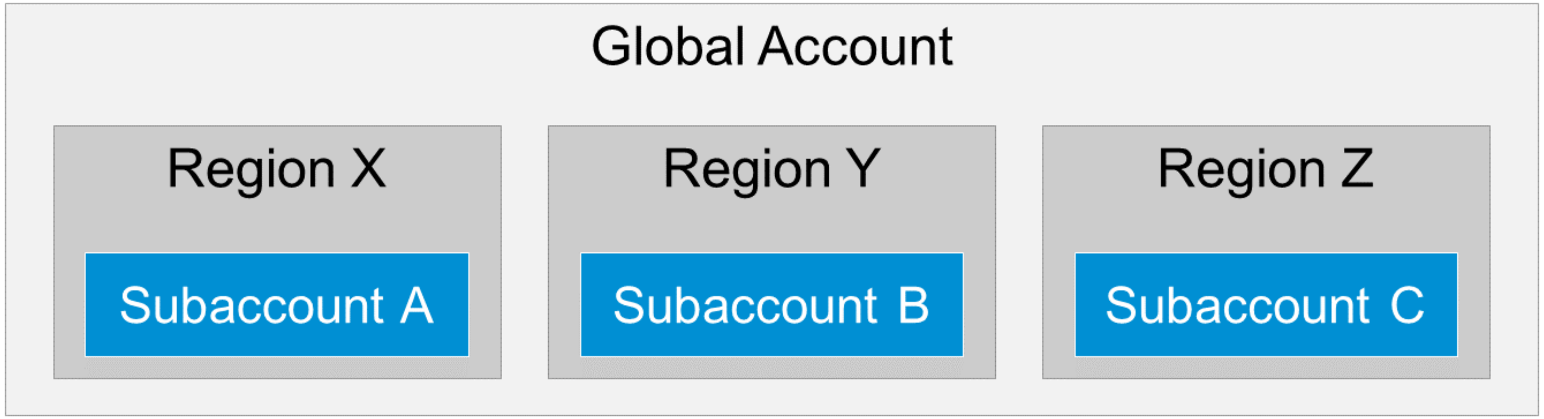
Steps involved

1. Initialize full-stack project
2. Create the tables – Data Modeling
3. Generic handlers – Out-of-the-box CRUD functionality
4. Basic UI
5. List Report layout
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

Steps involved

9. Consume external service in UI
10. Manual deployment to CF using manifest.yml
11. Manual deployment to CF using mta.yml
12. Security – Restrictions and Roles
13. Security – Authorization and Trust Management
14. Creating an AppRouter
15. Adding AppRouter to mat
16. CI / CD Pipeline

SAP BTP Account Model



Global Account

Region X

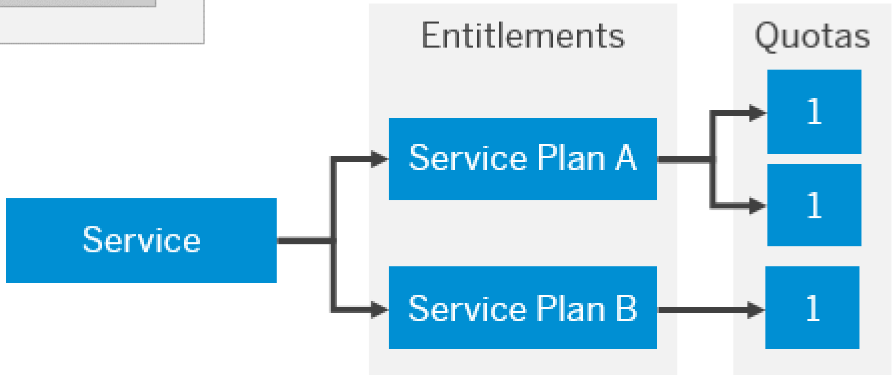
Subaccount A

Region Y

Subaccount B

Region Z

Subaccount C



Entitlements

Quotas

Service

Service Plan A

Service Plan B

1

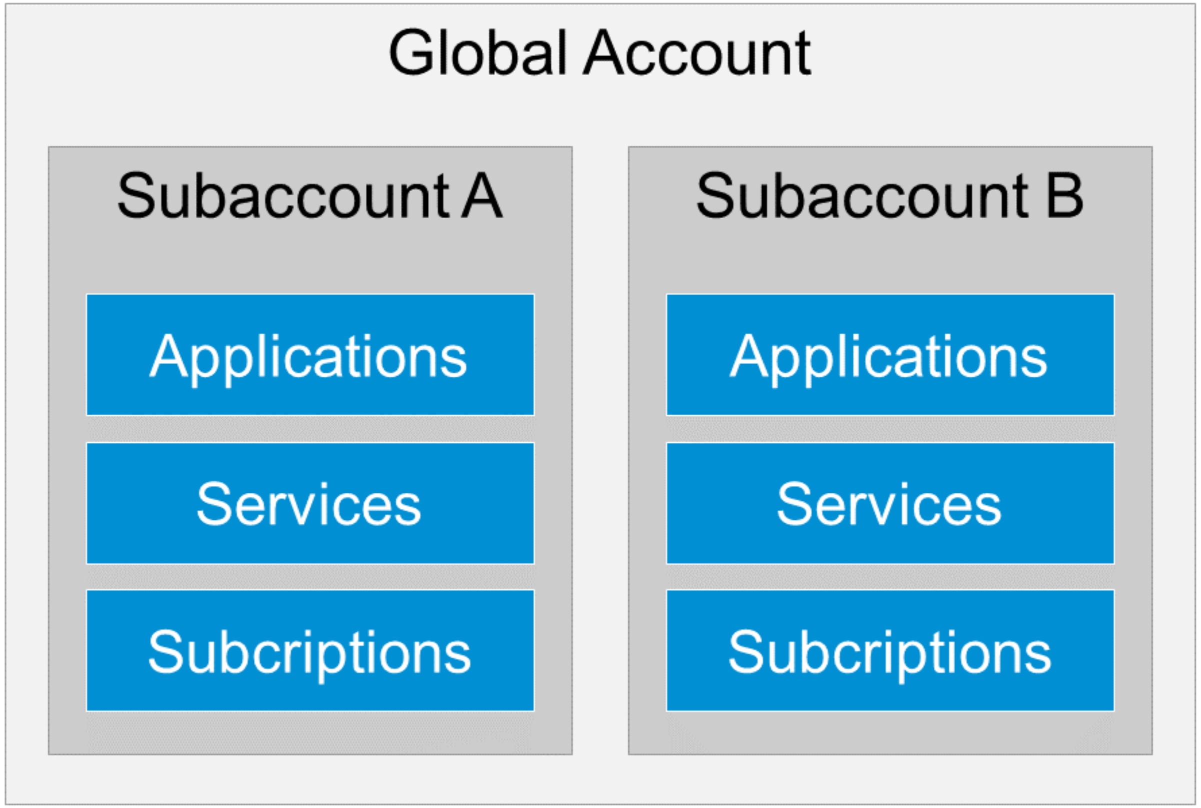
1

1

Global Accounts:

- When you sign a contract with SAP, you are provisioned with a Global Account
- Global account is used to manage subaccounts, entitlements, quotas etc.
  - For example, ABAP environment entitlement with 1 quota

SAP BTP Account Model



Global Account

Subaccount A

Applications

Services

Subcriptions

Subaccount B

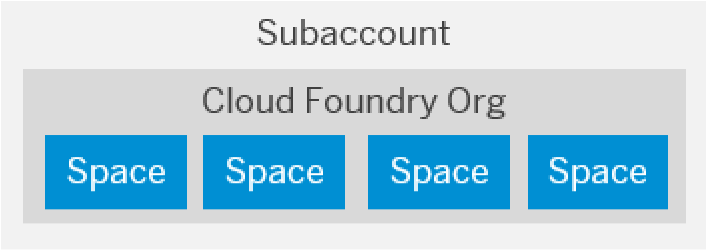
Applications

Services

Subcriptions

Subaccounts:

- Lets you structure your global account
- Subaccounts are independent of each other
- You can choose your own region
- Account model can be built on functional areas
    - Subaccount A – Sales and Marketing
    - Subaccount B - Development
    - Easy scaling
    - Reduced maintenance and governance efforts



Subaccount

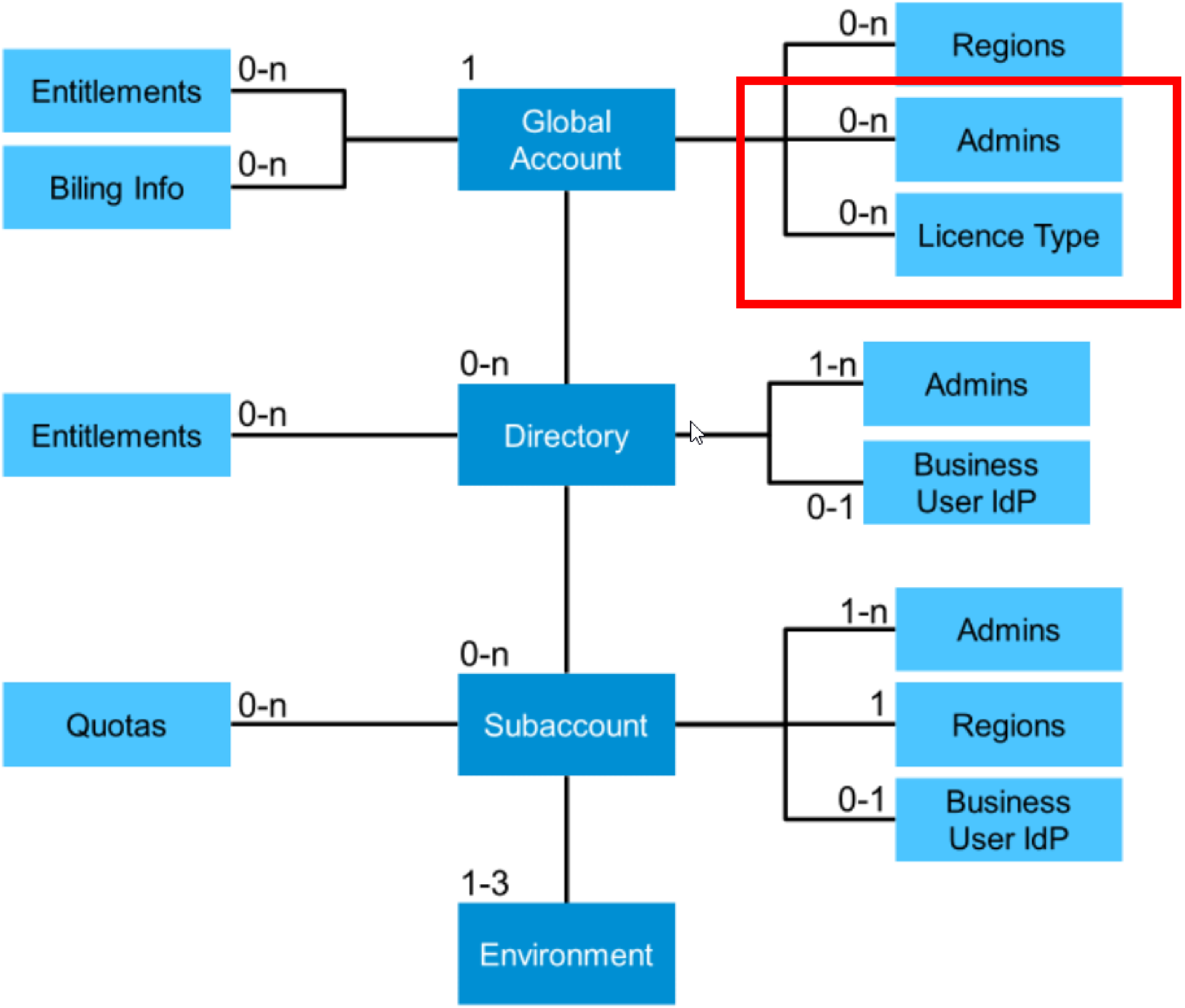
Cloud Foundry Org

Space

Space

Space

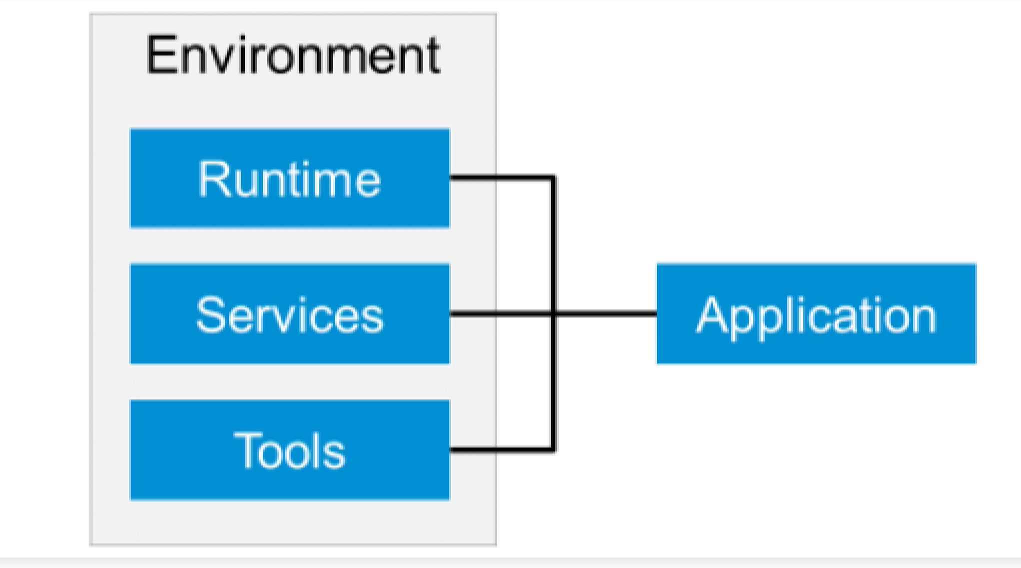
Space



There has to be at least 1 Admin…

Not sure about 0 License Type

Cloud Foundry Environment
ABAP Environment
Kyma Environment
Neo Environment



Environments constitute the actual platform-as-a-service offering of SAP BTP that allows for the development and administration of business applications.

Steps involved

1. Initialize full-stack project
2. Create the tables – Data Modeling
3. Generic handlers – Out-of-the-box CRUD functionality
4. Basic UI
5. List Report layout
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

Step 1 – Initialize Full-Stack project

```
git clone https://github.com/miltonchandradas/riskmanagement.git
```

```
cds init <Name of Project>
```

```
git checkout 0_initialize_project (Use tab for branch name)
```

Folders created

- app
- db
- srv

Files created

- package.json

Steps involved

1. Initialize full-stack project (Completed)
2. Create the tables – Data Modeling
3. Generic handlers – Out-of-the-box CRUD functionality
4. Basic UI
5. List Report layout
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

Step 2 – Create tables – Data Modeling

```
git checkout 1_data_modeling (Use tab for branch name)
```

Data Modeling Folder

- db

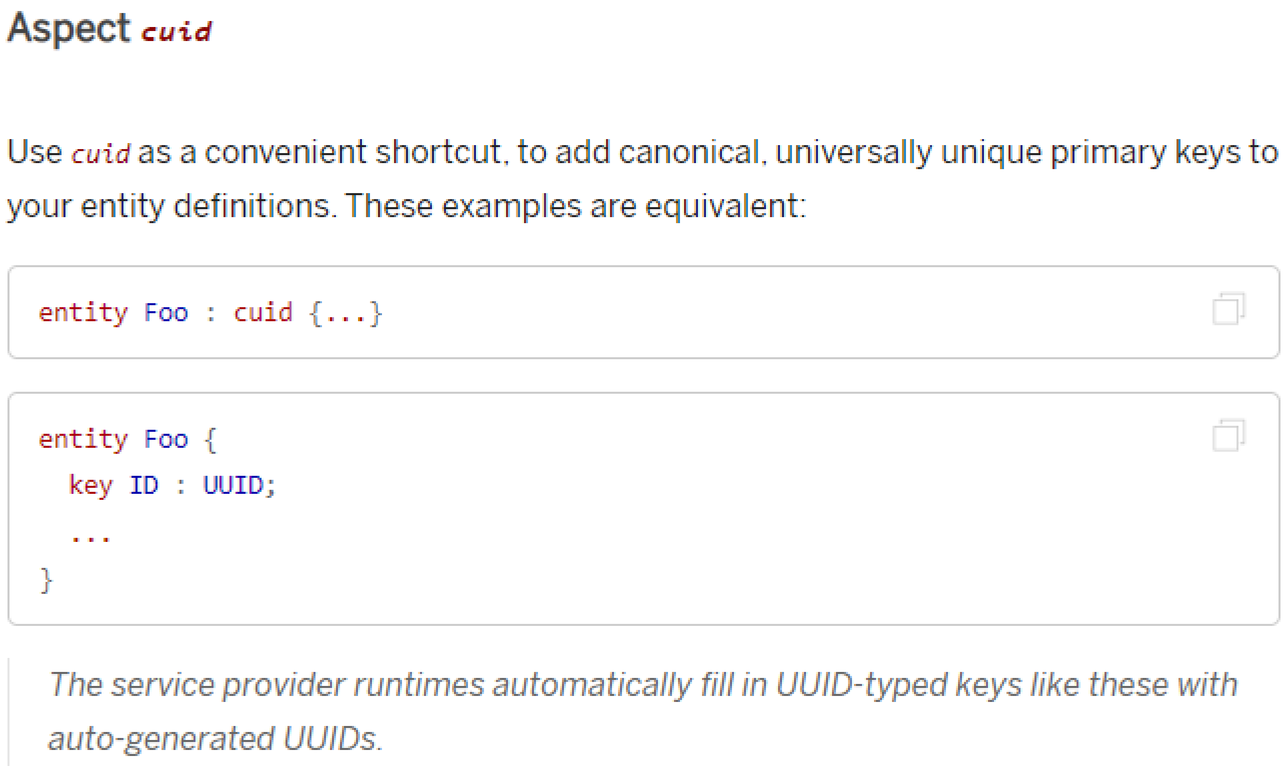
New Files

- schema.cds
- data/riskmanagement-Mitigations.csv
- data/riskmanagement-Risks.csv

Run application command

- cds watch

Step 2 – Create tables – Data Modeling



Aspect `cuid`

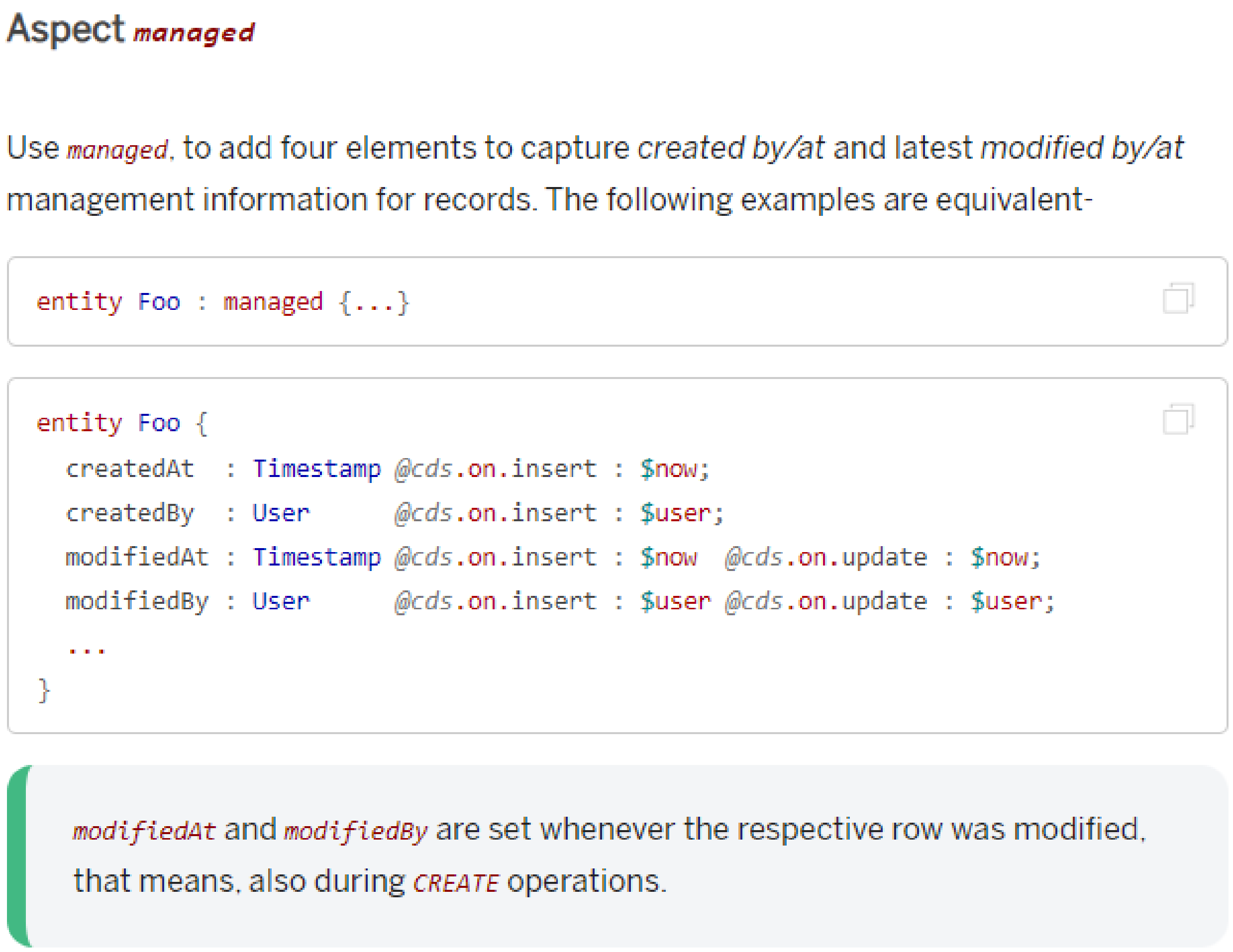
Use `cuid` as a convenient shortcut, to add canonical, universally unique primary keys to your entity definitions. These examples are equivalent:

```
entity Foo : cuid {...}
```

```
entity Foo {
  key ID : UUID;
  ...
}
```

> *The service provider runtimes automatically fill in UUID-typed keys like these with auto-generated UUIDs.*

Step 2 – Create tables – Data Modeling



Aspect *managed*

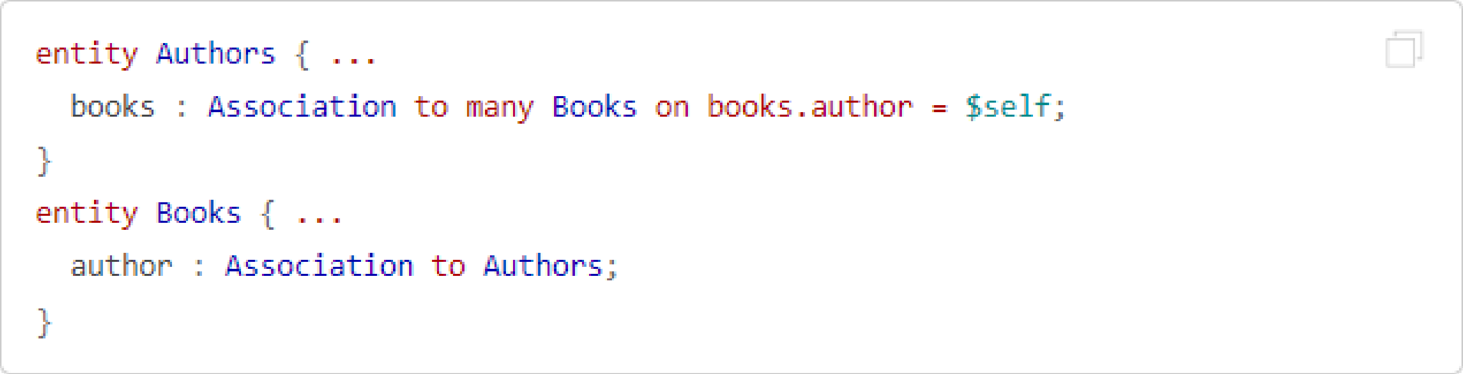
Use *managed*, to add four elements to capture *created by/at* and latest *modified by/at* management information for records. The following examples are equivalent-

```
entity Foo : managed {...}
```

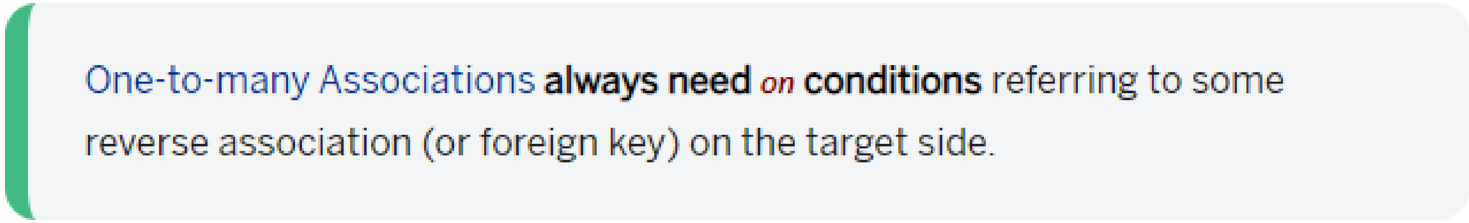
```
entity Foo {
  createdAt  : Timestamp @cds.on.insert : $now;
  createdBy  : User      @cds.on.insert : $user;
  modifiedAt : Timestamp @cds.on.insert : $now  @cds.on.update : $now;
  modifiedBy : User      @cds.on.insert : $user @cds.on.update : $user;
  ...
}
```

> *modifiedAt* and *modifiedBy* are set whenever the respective row was modified, that means, also during *CREATE* operations.

Step 2 – Create tables – Data Modeling



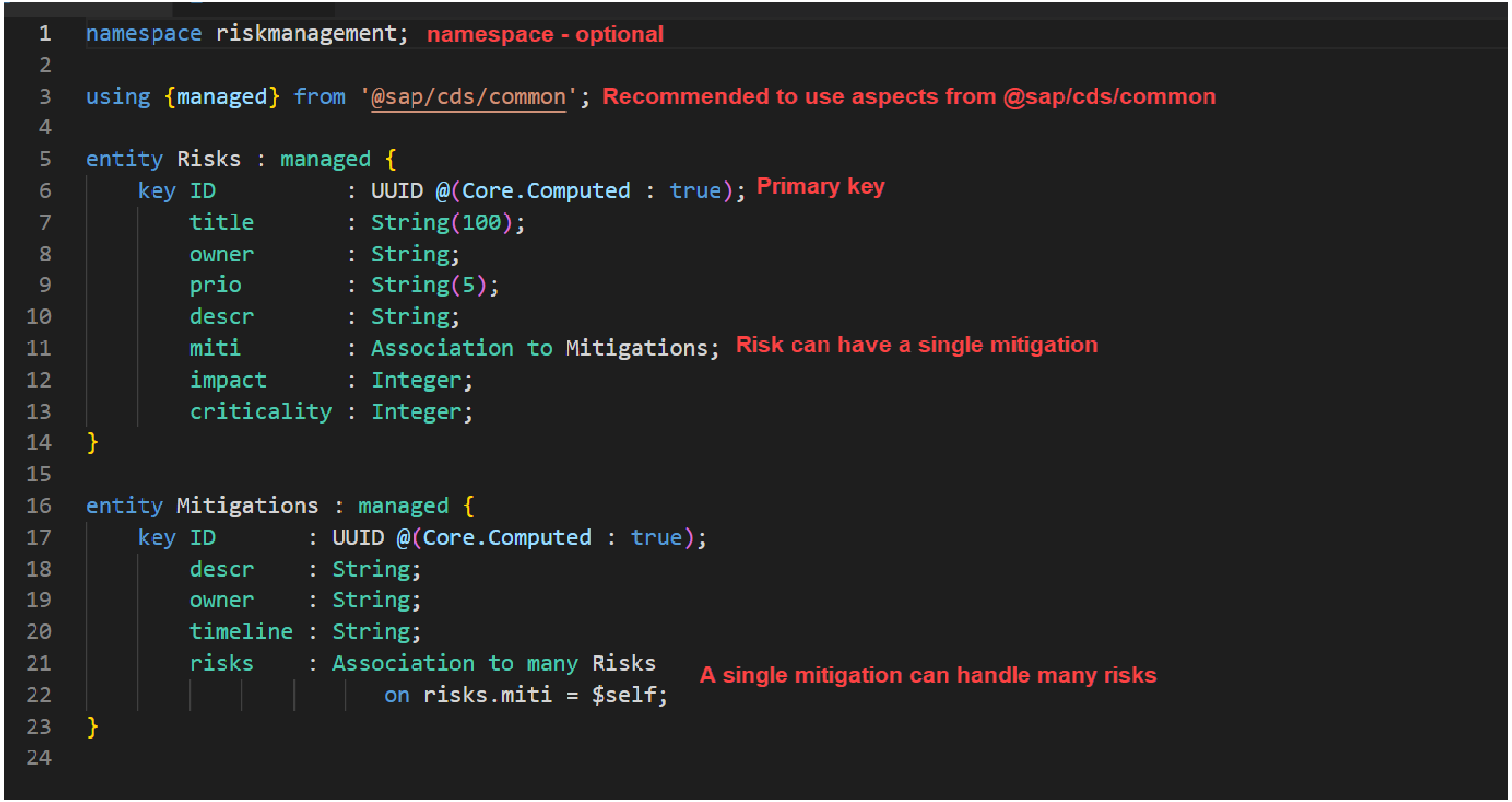
```
entity Authors { ...
  books : Association to many Books on books.author = $self;
}
entity Books { ...
  author : Association to Authors;
}
```



One-to-many Associations **always need** ***on*** **conditions** referring to some reverse association (or foreign key) on the target side.

Author can have MANY Books
Book can have 1 Author

Step 2 – Create tables – Data Modeling



```
namespace riskmanagement;  namespace - optional

using {managed} from '@sap/cds/common'; Recommended to use aspects from @sap/cds/common

entity Risks : managed {
    key ID          : UUID @(Core.Computed : true); Primary key
        title       : String(100);
        owner       : String;
        prio        : String(5);
        descr       : String;
        miti        : Association to Mitigations;  Risk can have a single mitigation
        impact      : Integer;
        criticality : Integer;
}

entity Mitigations : managed {
    key ID       : UUID @(Core.Computed : true);
        descr    : String;
        owner    : String;
        timeline : String;
        risks    : Association to many Risks
                       on risks.miti = $self;      A single mitigation can handle many risks
}
```

Steps involved

1. Initialize full-stack project - Completed
2. Create the tables – Data Modeling - Completed
3. Generic handlers – Out-of-the-box CRUD functionality
4. Basic UI
5. List Report layout
6. Custom event handling – Business logic
7. Support for external API
8. Connecting to Sandbox

Step 3 – Generic Handlers

`git checkout 2_generic_handlers (Use tab for branch name)`

Service Folder

- srv

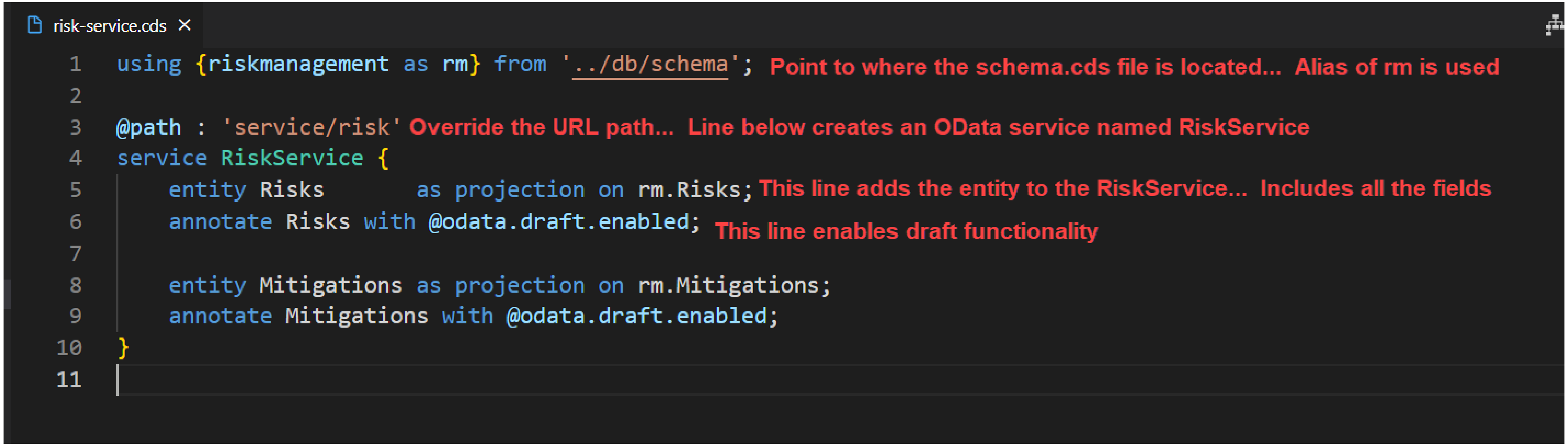
New Files

- risk-service.cds

Run application command

- cds watch

Step 3 – Generic Handlers



```
using {riskmanagement as rm} from '../db/schema';

@path : 'service/risk'
service RiskService {
    entity Risks       as projection on rm.Risks;
    annotate Risks with @odata.draft.enabled;

    entity Mitigations as projection on rm.Mitigations;
    annotate Mitigations with @odata.draft.enabled;
}

```

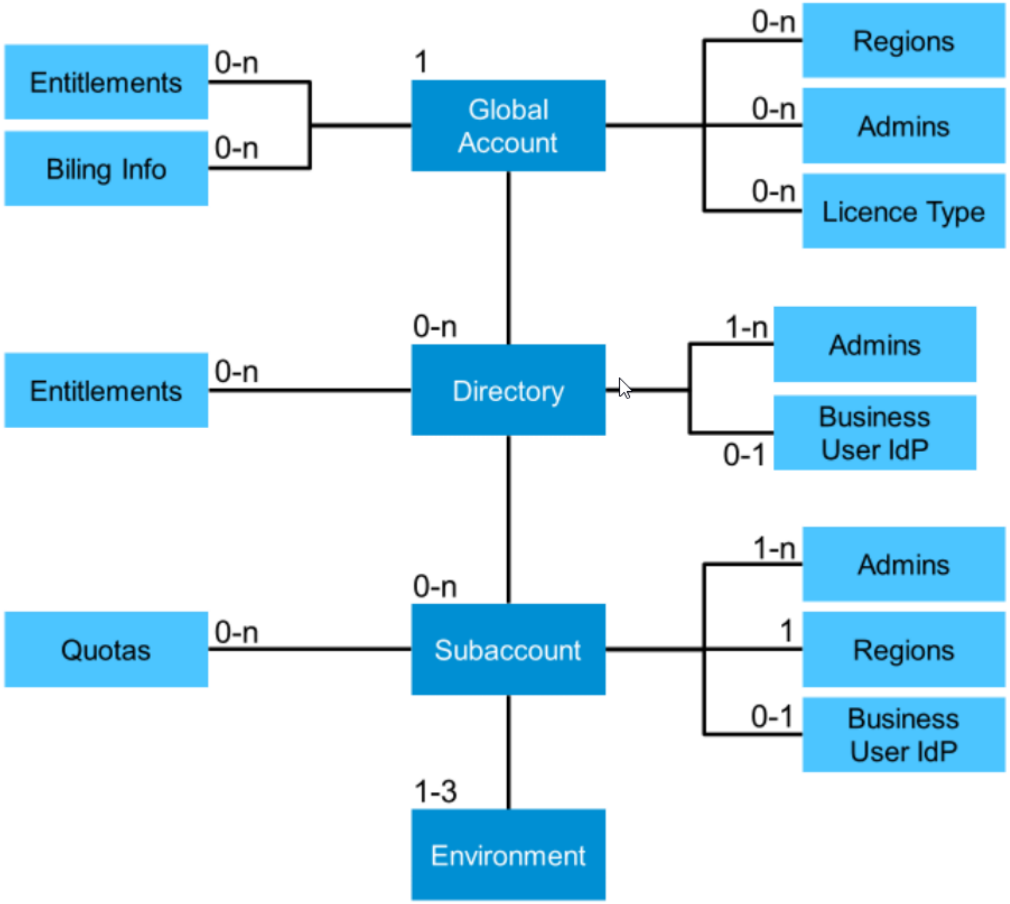
Point to where the schema.cds file is located... Alias of rm is used

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

Key Summary Points – Unit 1, 2

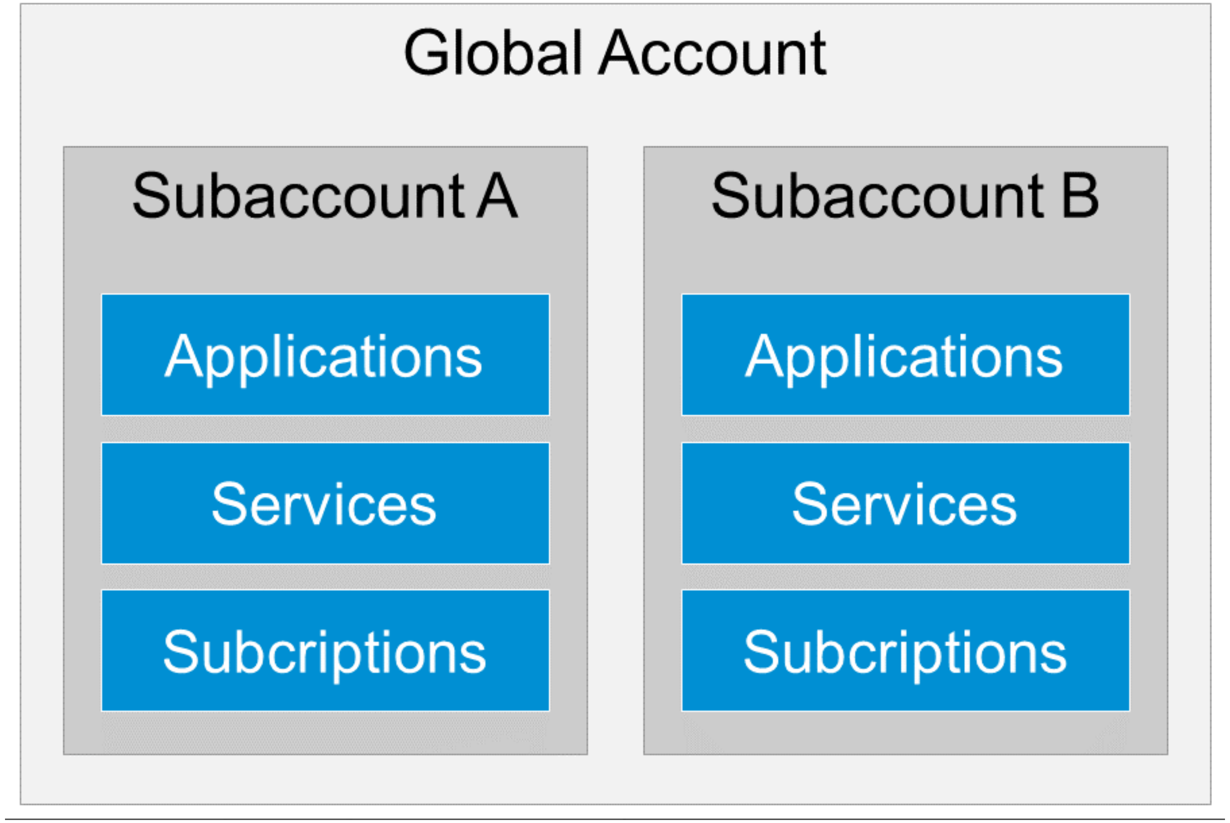


Key Summary Points – Unit 1, 2

Environments available in SAP BTP for building applications

- Kyma Environment
- Cloud Foundry Environment
- ABAP Environment
  - Technically, the ABAP environment lives within the Cloud Foundry (CF) environment
  - Supports ABAP RESTful Application Programming Model (RAP)
  - Based on the latest ABAP platform cloud release

Key Summary Points – Unit 1, 2



Subaccounts:

- Lets you structure your global account
- Subaccounts are independent of each other
- You can choose your own region
- Account model can be built on functional areas
  - Subaccount A – Sales and Marketing
  - Subaccount B - Development
  - Easy scaling
  - Reduced maintenance and governance efforts

Key Summary Points – Unit 1, 2

```
cds init <Name of Project>
```

Folders created

- app
- db
- srv

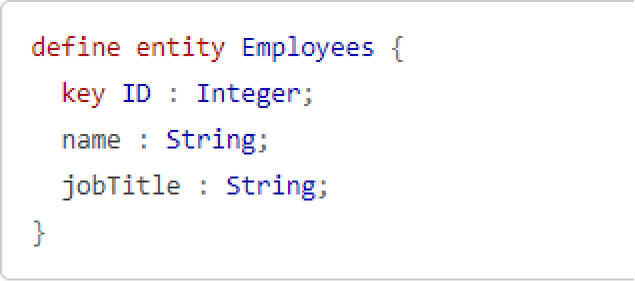
Files created

- package.json

Key Summary Points – Unit 1, 2

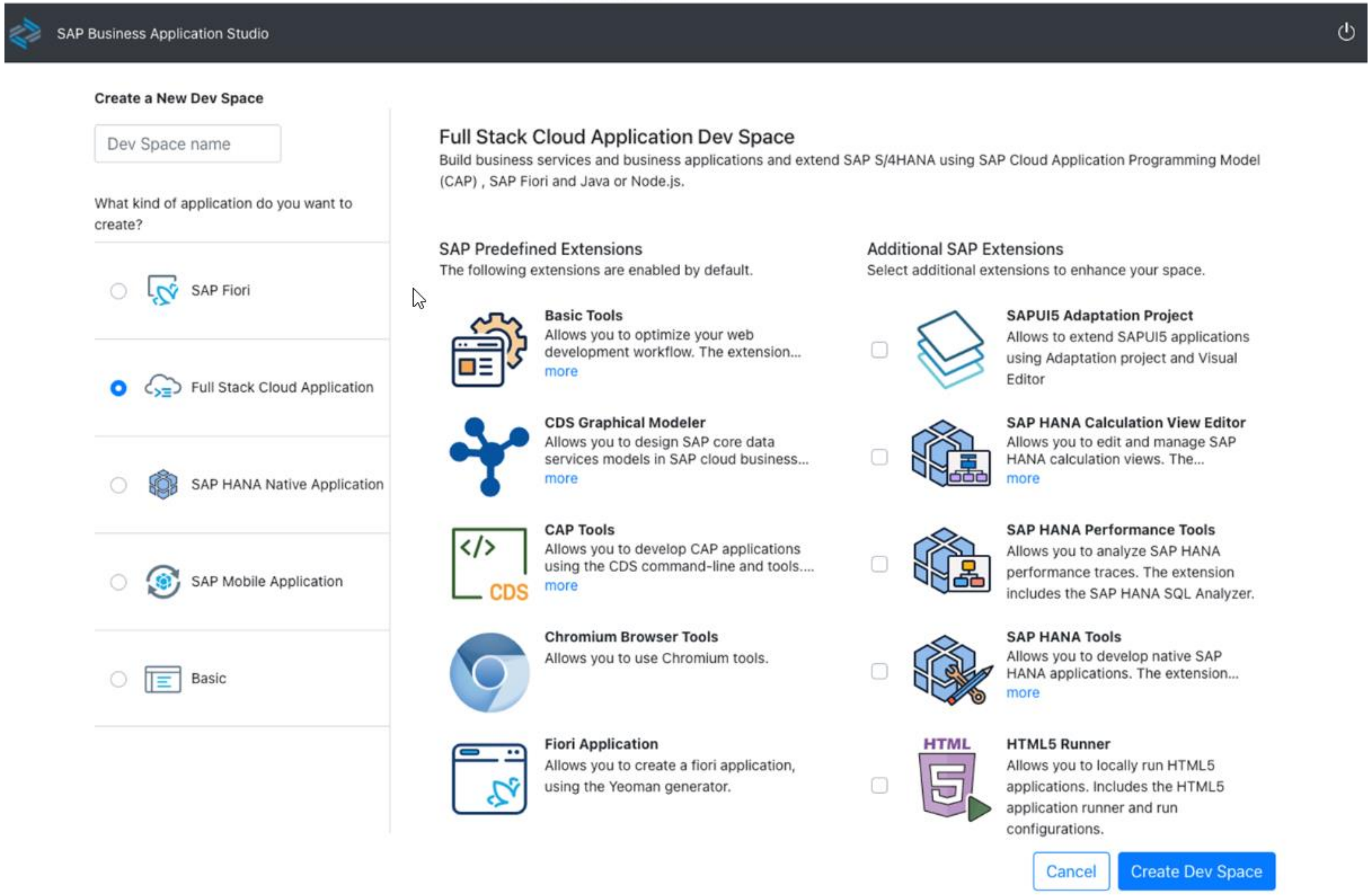
Core Data Service Definition Language (CDL)

- Human readable syntax for defining data models



```
define entity Employees {
  key ID : Integer;
  name : String;
  jobTitle : String;
}
```

Key Summary Points – Unit 1, 2



Create a New Dev Space

Dev Space name

What kind of application do you want to create?

○ SAP Fiori

● Full Stack Cloud Application

○ SAP HANA Native Application

○ SAP Mobile Application

○ Basic

Full Stack Cloud Application Dev Space

Build business services and business applications and extend SAP S/4HANA using SAP Cloud Application Programming Model (CAP) , SAP Fiori and Java or Node.js.

SAP Predefined Extensions

The following extensions are enabled by default.

Basic Tools
Allows you to optimize your web development workflow. The extension... more

CDS Graphical Modeler
Allows you to design SAP core data services models in SAP cloud business... more

CAP Tools
Allows you to develop CAP applications using the CDS command-line and tools.... more

Chromium Browser Tools
Allows you to use Chromium tools.

Fiori Application
Allows you to create a fiori application, using the Yeoman generator.

Additional SAP Extensions

Select additional extensions to enhance your space.

□ **SAPUI5 Adaptation Project**
Allows to extend SAPUI5 applications using Adaptation project and Visual Editor

□ **SAP HANA Calculation View Editor**
Allows you to edit and manage SAP HANA calculation views. The... more

□ **SAP HANA Performance Tools**
Allows you to analyze SAP HANA performance traces. The extension includes the SAP HANA SQL Analyzer.

□ **SAP HANA Tools**
Allows you to develop native SAP HANA applications. The extension... more

□ **HTML5 Runner**
Allows you to locally run HTML5 applications. Includes the HTML5 application runner and run configurations.

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

- Service Document
- Service Metadata Document

Key Summary Points – Unit 1, 2

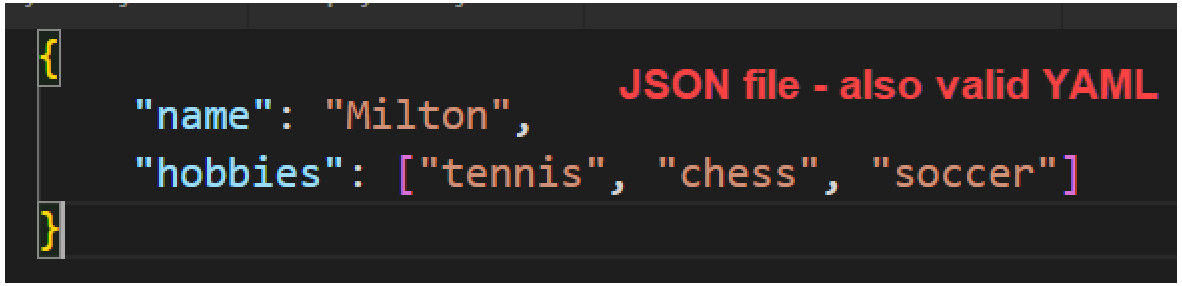
JSON

- 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

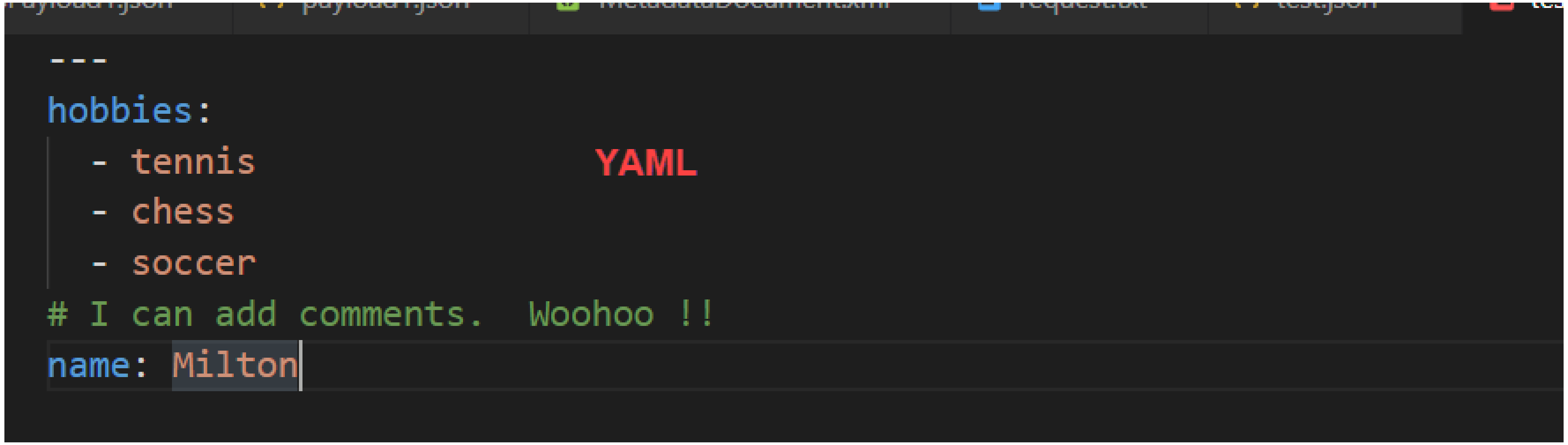
- Unicode based data serialization language
- YAML is a strict JSON superset – this means all valid JSON files are valid YAML files
- Support for serializing arbitrary native data structures

Key Summary Points – Unit 1, 2



JSON file - also valid YAML

```json
{
    "name": "Milton",
    "hobbies": ["tennis", "chess", "soccer"]
}
```



YAML

```yaml
---
hobbies:
  - tennis
  - chess
  - soccer
# I can add comments.  Woohoo !!
name: Milton
```