Stackml workshop

Practical work

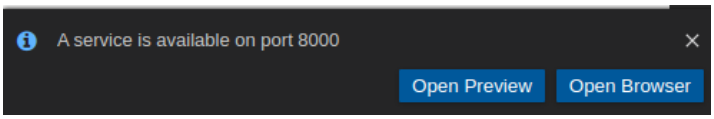
Objectif : l'objectif de cette série d'exercices est de t'aider à décrire, en stackml, une architecture de système IoT bout en bout, pas à pas, en suivant la méthodologie TENPA.
A la fin des 90 minutes tu devrais avoir finalisé les 7 steps et constitué:

- Le fichier principal de ton architecture
- Les fichiers de bibliothèques d'entités que tu importes
- Le diagramme général du système
- La topologie de deploiement
- La carte du déploiement
- Le diagramme en couches du système

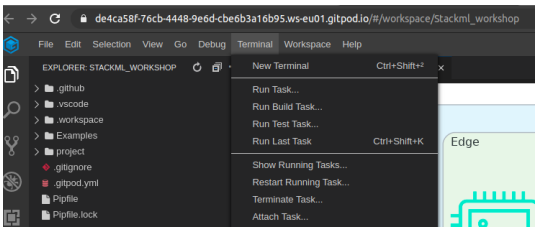
La documentation est accessible en ligne: https://stackml.stackeo.io/index.html

Step #1 : Setup

1. Se connecter sur Gitpod
2. Aller à https://gitpod.io/#https://github.com/Stackeo-io/Stackml_workshop
3. Cliquer sur Open Preview



4. Ouvrir un nouveau terminal



5. Installer stackml : `pip install stackml`
6. Tester l'exemple:
    a. Visualiser le projet Examples/Test
    b. Checker le projet : stackml check -i Examples/Test
    c. Introduire une erreur syntaxique et checker puis corriger
    d. Générer un image générale du système:

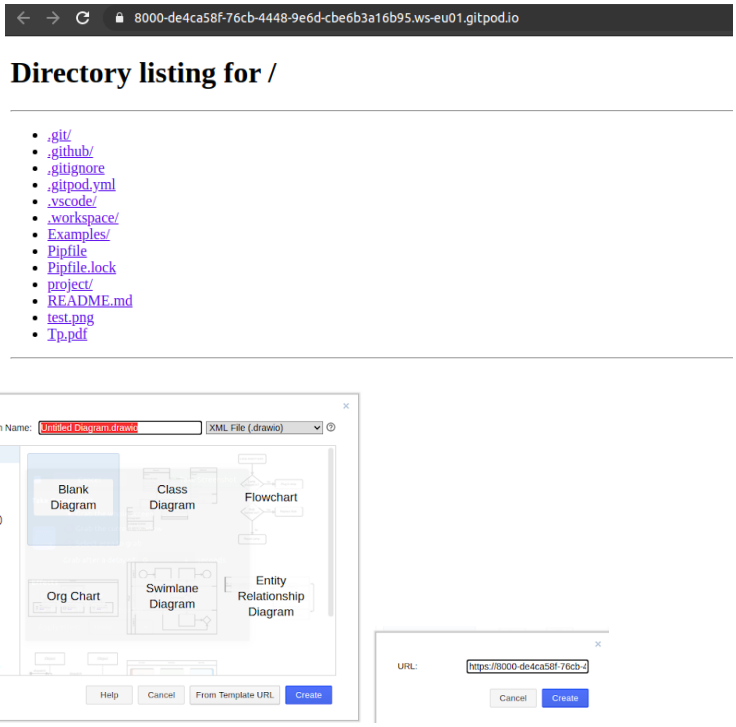
```
stackml compile -i Examples/Test diagram -t 1 -o diagramme1
```

- ouvrir en cliquant sur le fichier ou avec la preview web (actualiser la page)

e. Générer un diagramme du système, editable:

```
 stackml compile -i Examples/Test drawio -l 1 -o diagramme1  -i
Examples/Test/icons/
```

- ouvrir avec https://app.diagrams.net/ en utilisant le URL du fichier drawio à partir de la preview



f. Générer un diagramme editable des couches du système (niveau 2) :

```
stackml compile -i Examples/Test drawio -l 2 -o diagramme2
```

- ouvrir avec https://app.diagrams.net/ en utilisant le URL du fichier drawio à partir de la preview

g. Générer la topologie de déploiement:

```
stackml compile -i Examples/Test diagram -t 2 -o diagramme3
```

- ouvrir en cliquant sur le fichier ou avec l'interface web

h. Générer la carte de déploiement:

```
stackml compile -i Examples/Test diagram -t 3 -o diagramme4 -i
```

```
Examples/Test/icons/
```

- ouvrir avec l'interface web

Step #2 : Preparation du Use Case

1. Choisir et préparer votre use case (il vous faut un sketch de son architecture)
2. Identifier la liste des tiers requis parmi : Things, Edge, Network, Platform, Application
3. Lister l'inventaire les modèles de nodes nécessaires.
4. Rechercher si les bibliothèques de modèles de nodes requis existent, sinon, il vous faudra créer les fichiers correspondants (voir Step4).
5. Réfléchir aux liens entre ces nodes.

Step #3 : Création de la topologie logique (niveau 1)

> ➢ Rajouter toujours l'extension .stkml.yaml au fichiers stackml
> ➢ Utiliser ctrl+space pour l'auto-complétion et les suggestions

1. Développer le fichier principal (main) de votre use case
    a. Créer un nouveau répertoire
    b. Stackml init (éventuellement -p nomrep - # pour créer un nouveau répertoire)
    c. Ouvrir le fichier main
    d. Pour définir l'architecture haut niveau du système de votre use case:
        i. Importer les bibliothèques de modèles de nodes requises dans votre topologie. Si elles n'existent pas passer au step #4 suivant puis revenir ici.
        ii. Décrire la topologie logique en lui donnant un nom et une catégorie de useCase, listant les nodes (id, model, link) puis les links entre ces nodes (source & sink)
2. Checker le fichier
3. Generer le diagram type 1 de diagram et drawio .

Step #4 : création des bibliothèques de modèles de nodes (niveau1)

1. Développer les fichiers à importer dans votre use case (niveau1) (rajouter l'extension stkml.yaml pour bénéficier de l'auto-complétion et la vérification syntaxique instantanée)
2. Développer le fichier des modèles de nodes de capteurs, de devices (tier Thing), de gateway (tier Edge) (par exemple Kontron), de nuage réseau (tier Network), de nodes de traitement ou de stockage de données (tier Platform), de nodes d'applications (tier Application)
3. Par exemple pour un fichier de modèle de edge gateway, créer le fichier Kontron.stkml.yaml
4. Exemple pour définir le modèle KBoxA203

```
modeldef:
    - Node:
      id: KBoxA203
      type: EdgeNode
```

Step #5 : Définir le déploiement : regions et populations

1. Détailler le fichier main pour définir les régions dans la section en dessous de la section Topologie, puis dans chaque région, définir les populations de nodes a déployer.
2. Introduire la section Régions (name, type) et pour chacune détailler le nombre de nodes (population).
3. Checker le fichier
4. Générer la carte (map) du déploiement.

Step #6 : Modélisation en couche de la topologie (niveau2)

1. Détailler le fichier main pour définir les layer.
2. Importer les bibliothèques détaillées correspondantes à votre use case. Si elles ne sont pas détaillées passer au step 6.
3. Pour définir le niveau 2 du système, c'est à dire les layers éléments au sein de chaque node et les liens entre les composants des nodes, il suffit d'ajouter le nom des layers correspondantes aux différentes relations de votre use case comme attribut des links.
4. Checker le fichier
5. Générer le diagram drawio correspondent (-l 2)

Step #7 : Détailler les bibliothèques de modèles de nodes (niveau2)

1. Détailler les fichiers à importer dans votre use case (niveau 2)

2. Reprendre les fichiers des modèles de Things, de Edge (par exemple Kontron), de Network, de Platform, d'Applications
3. Pour chacun, il faudra, le cas échéant, décrire
    a. la layer energy
    b. la layer physical avec ses composants
    c. layer network avec ses composants
    d. la layer connectivity
    e. la layer data

Par exemple pour la définition de la box de capteur Lyra:

```
modeldef:
    - NodeModels:
        id: Lyra
        type: ThingNode
        EnergylayerElement:
          - ComponentId: lyraPower
            nature: Hardware
        PhysicallayerElement:
          - ComponentId: lyraHardware
            nature: Hardware
          - ComponentId: CoreSensor
            nature: Hardware
        NetworklayerElement:
          - ElementType: EndPoint
          - ComponentId: lyraBle
            nature: Hardware
        ConnectivitylayerElement:
          - ElementType: EndPoint
          - ComponentId: lyraSSL
            nature: Software
        DatalayerElement:
          - ElementType: EndPoint
          - ComponentId: lyraDataCollect
            nature: Software
```