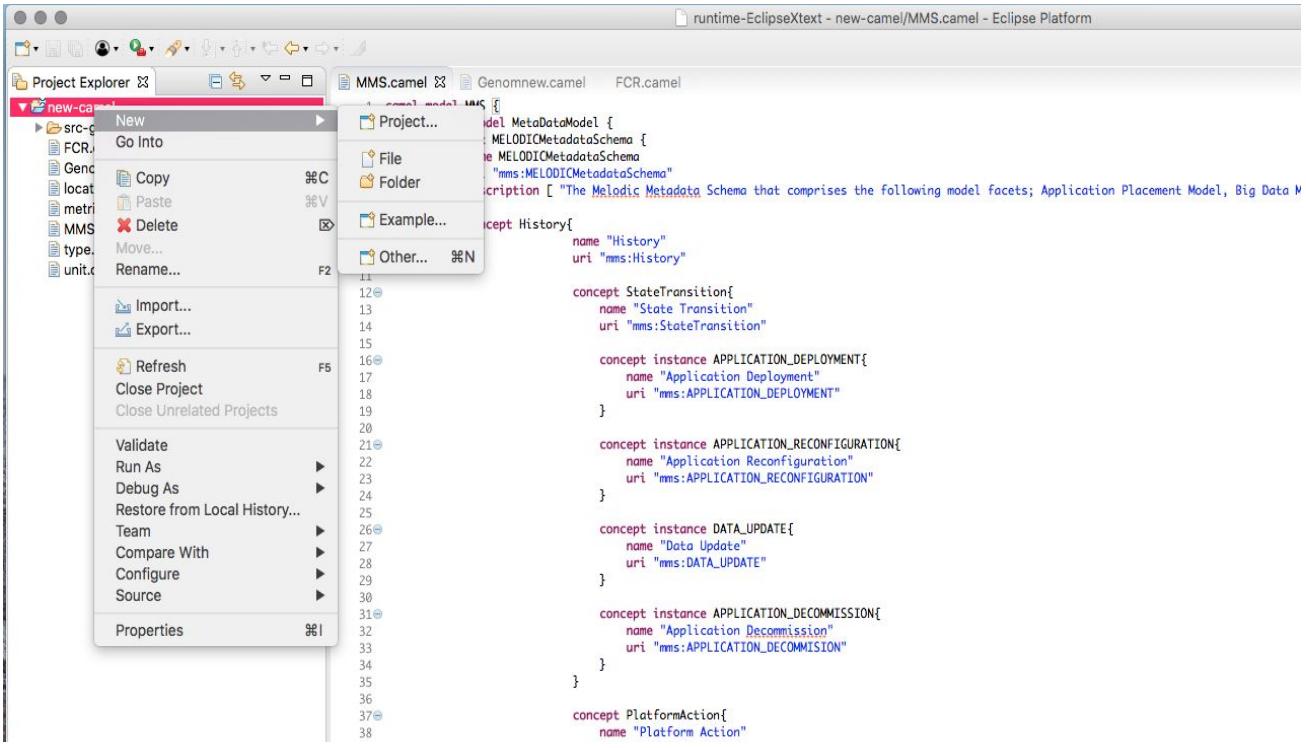
Modeling and deployment your own application in CAMEL

1. Essential information for preparing CAMEL model

- CAMEL FAQ
- Deployment Modelling
- Requirements Modelling
- Resource Requirements
- Data Modelling
- Value Type Modelling
- CAMEL web site

2. Open camel editor (installed and configured in scenario 3) and create new file named with .camel extension.



3. Example of two component application CAMEL model, which you can use it as a based for your model. Green comments provides helpful information.

```
camel model TwoComponentApp{
    application TwoComponentApp{
        version '2.5'
    }
    deployment type model TwoComponentAppDeployment{
    //decribed in Deployment Modelling (point 1)
        software Component_App{
            requirements AppRequirementSet

            script configuration ComponentAppConfiguration{
                    // script configuration set installation script for the VM
                download 'rm -rf ~/melodic && mkdir ~/melodic && cd ~/melodic && wget
https://s3-eu-west-1.amazonaws.com/melodic.testing.data/APP.sh && chmod +x
~/melodic/APP.sh'
                install '~/melodic/APP.sh install'
                configure 'mkdir ~/test2'
                start 'printenv >> ~/melodic/env.txt && ~/melodic/APP.sh start'
            }
            provided communication ComponentAppPort port 9999
            required communication ComponentPortDbReq port 3306 mandatory

        }
        software Component_DB{

[MetaDataModel.MELODICMetadataSchema.UtilityNotions.UtilityRelatedProperties.Unmoveable]
            //once the component is deployed it is not possible to move it

            requirements DBRequirementSet
            required host WM_DBHostReq
            script configuration ComponentDBConfiguration{
                download 'rm -rf ~/melodic && mkdir ~/melodic && cd ~/melodic && wget
https://s3-eu-west-1.amazonaws.com/melodic.testing.data/MySQLDB.sh && chmod +x
~/melodic/MySQLDB.sh'
                install '~/melodic/MySQLDB.sh install'
                configure '~/melodic/MySQLDB.sh configure'
                start '~/melodic/MySQLDB.sh start'
            }
            provided communication ComponentDBPort port 3306
            longLived
        }

        communication AppToDB from Component_App.ComponentPortDbReq to
Component_DB.ComponentDBPort
        //communication rules between components

        requirements AppRequirementSet{
             //invoking requirements defined in requirement model
            resource TwoComponentApp_Requirement.AppReqs
            horizontal scale
TwoComponentApp_Requirement.HorizontalScaleTwoComponentAppWApp
            image TwoComponentApp_Requirement.Ubuntu
        }
        requirements DBRequirementSet{
            resource TwoComponentApp_Requirement.DBReqs
            horizontal scale
TwoComponentApp_Requirement.HorizontalScaleTwoComponentAppDB
            image TwoComponentApp_Requirement.Ubuntu
        }
    }
    requirement model TwoComponentApp_Requirement{
    //decribed in Requirements Modelling and Resource Requirements

        resource requirement AppReqs{
             //requirements for cores and RAM, need to be indicated for each component
            feature coresApp{

[ MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.CPU]
                attribute minCoresApp
```

```
[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.CPU.hasMi
nNumberofCores]
                : int 1
                attribute maxCoresApp

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.CPU.hasMa
xNumberofCores]
                : int 8
            }
            feature ramApp{

[ MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.RAM]
                attribute minRamApp

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.RAM.Total
Memory.totalMemoryHasMin ]
                : int 8100
                attribute maxRamApp

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.RAM.Total
Memory.totalMemoryHasMax]
                : int 10072
            }
        }
        resource requirement DBReqs{
            feature coresDB{

[ MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.CPU]
                attribute minCoresDB

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.CPU.hasMi
nNumberofCores]
                : int 1
                attribute maxCoresDB

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.CPU.hasMa
xNumberofCores]
                : int 2
            }
            feature ramDB{

[ MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.RAM]
                attribute minRamDB

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.RAM.Total
Memory.totalMemoryHasMin ]
                : int 1024
                attribute maxRamDB

[MetaDataModel.MELODICMetadataSchema.ApplicationPlacementModel.IaaS.Processing.RAM.Total
Memory.totalMemoryHasMax]
                : int 3100
            }
        }
             //number of instances
        horizontal scale requirement HorizontalScaleTwoComponentAppWApp [1,1]
        horizontal scale requirement HorizontalScaleTwoComponentAppDB [1,1]
             //set name of image for the VMs (first value in AWS's public AMI for
ubuntu18 and the second is Openstack's )
             image requirement Ubuntu  [ 'ubuntu-bionic-18.04-amd64-server-20190627.1-
disabled-unattented-upgrades','ubuntu-1604' ]
    }
}
```

4. Model you own application, using example from point 3, containing:

- two dockerized component: database (imageId: mariadb) application (imageId: wordpress),
- defined communication rules between components,
- resource requirement:
  - database : 1-2 cores, 1024 - 3100 RAM
  - application : 1-2 cores, 8100 - 3100 RAM
- image requirement for both components ubuntu-1604

HINT! ctrl+space - gives you a prompt

5. Compile CAMEL file (ctrl+s) and you should have you .xmi file ready to deploy

6. Based on scenario 2, deploy your own application using Melodic platform