
MICO Documentation

Release 0.0.1

MICO Authors

Oct 06, 2019

Contents:

| | | |
|----------|--|-----------|
| 1 | Mockups | 3 |
| 1.1 | Initial sketch of MICO | 3 |
| 2 | Domain Model | 11 |
| 2.1 | MicoApplication | 11 |
| 2.2 | MicoApplicationJobStatus | 11 |
| 2.3 | MicoEnvironmentVariable | 12 |
| 2.4 | MicoServiceBackgroundJob | 12 |
| 2.5 | KubernetesDeploymentInfo | 12 |
| 2.6 | MicoLabel | 12 |
| 2.7 | MicoMessage | 13 |
| 2.8 | MicoService | 13 |
| 2.9 | MicoServiceDependency | 14 |
| 2.10 | MicoServiceDeploymentInfo | 14 |
| 2.11 | MicoServiceInterface | 15 |
| 2.12 | MicoServicePort | 15 |
| 2.13 | MicoPortType | 15 |
| 2.14 | MicoServiceCrawlingOrigin | 15 |
| 2.15 | MicoVersion | 15 |
| 2.16 | MicoInterfaceConnection | 16 |
| 2.17 | MicoTopic | 16 |
| 2.18 | MicoTopicRole | 16 |
| 3 | Database Schema | 19 |
| 3.1 | Overview | 19 |
| 3.2 | Service | 19 |
| 3.3 | DependsOn | 20 |
| 3.4 | Service Interface | 20 |
| 3.5 | Application | 20 |
| 3.6 | Possible Nodes and Relationships | 21 |
| 4 | Messaging | 23 |
| 4.1 | CloudEvents | 23 |
| 4.2 | Error Report Messages | 25 |
| 5 | REST API | 27 |
| 5.1 | Structure Overview | 27 |

| | | |
|----------|--|------------|
| 5.2 | Openapi Specification | 28 |
| 5.3 | Method Details (specification) | 63 |
| 6 | Setup | 67 |
| 6.1 | MICO Setup | 67 |
| 6.2 | Azure Kubernetes Service | 70 |
| 6.3 | Google Kubernetes Engine | 74 |
| 6.4 | Development environment | 76 |
| 6.5 | CI/CD Pipeline | 87 |
| 6.6 | Kubernetes | 90 |
| 7 | Debugging & Testing | 119 |
| 7.1 | Testing | 119 |
| 7.2 | Debbuging | 124 |
| 7.3 | Postman | 124 |
| 8 | MICO Tutorials | 127 |
| 8.1 | How To add a service | 127 |
| 8.2 | How To manage a service | 131 |
| 8.3 | How To add an application | 135 |
| 8.4 | How To manage an application | 137 |
| 8.5 | How To create a messaging based application | 146 |
| 9 | Architectural Decision Log | 161 |
| 9.1 | [short title of solved problem and solution] | 161 |
| 9.2 | Use Markdown Architectural Decision Records | 163 |
| 9.3 | Java Framework | 163 |
| 9.4 | Performance considerations | 164 |
| 9.5 | Security considerations | 164 |
| 9.6 | Monolith as Architectural Style | 164 |
| 9.7 | Repository Structure Frontend/Backend | 165 |
| 9.8 | Web-UI Framework | 165 |
| 9.9 | Database Technology (Relational/Graph) | 166 |
| 9.10 | Browser Compatibility | 168 |
| 9.11 | Features first | 168 |
| 9.12 | REST API Design | 169 |
| 9.13 | JSON+hal as serialization Format for the API | 169 |
| 9.14 | Re-use capabilites of Winery Topology Modeler | 170 |
| 9.15 | Source-to-Image Workflow | 171 |
| 9.16 | Evaluate Spring Data REST | 173 |
| 9.17 | Building OCI Images | 174 |
| 9.18 | Documentation Structure | 176 |
| 9.19 | Istio vs. Envoy | 178 |
| 9.20 | Requirements regarding the Composition of Applications | 179 |
| 9.21 | Architecture for background jobs | 180 |
| 9.22 | Configurable Service Dependencies | 181 |
| 9.23 | Kafka as Messaging Middleware | 182 |
| 9.24 | Function to component mapping | 183 |
| 9.25 | FaaS | 185 |
| 9.26 | Language for a generic composition pattern implementation | 187 |
| 9.27 | [WIP] Kafka-faas-connector requirements (aka generic component requirements) | 187 |
| 9.28 | Implementation of complex EAI-Patterns with FaaS | 190 |
| 9.29 | Simple Composition Components | 193 |
| 9.30 | Open Decisions | 194 |

| | |
|---------------------------|------------|
| 10 Glossary | 195 |
| HTTP Routing Table | 197 |
| Index | 199 |

Mico is a project for designing, deploying and managing *applications* consisting of multiple *services*.

See also:

[MICO Developer Documentation](#)

[Documentation Structure](#)

CHAPTER 1

Mockups

1.1 Initial sketch of MICO

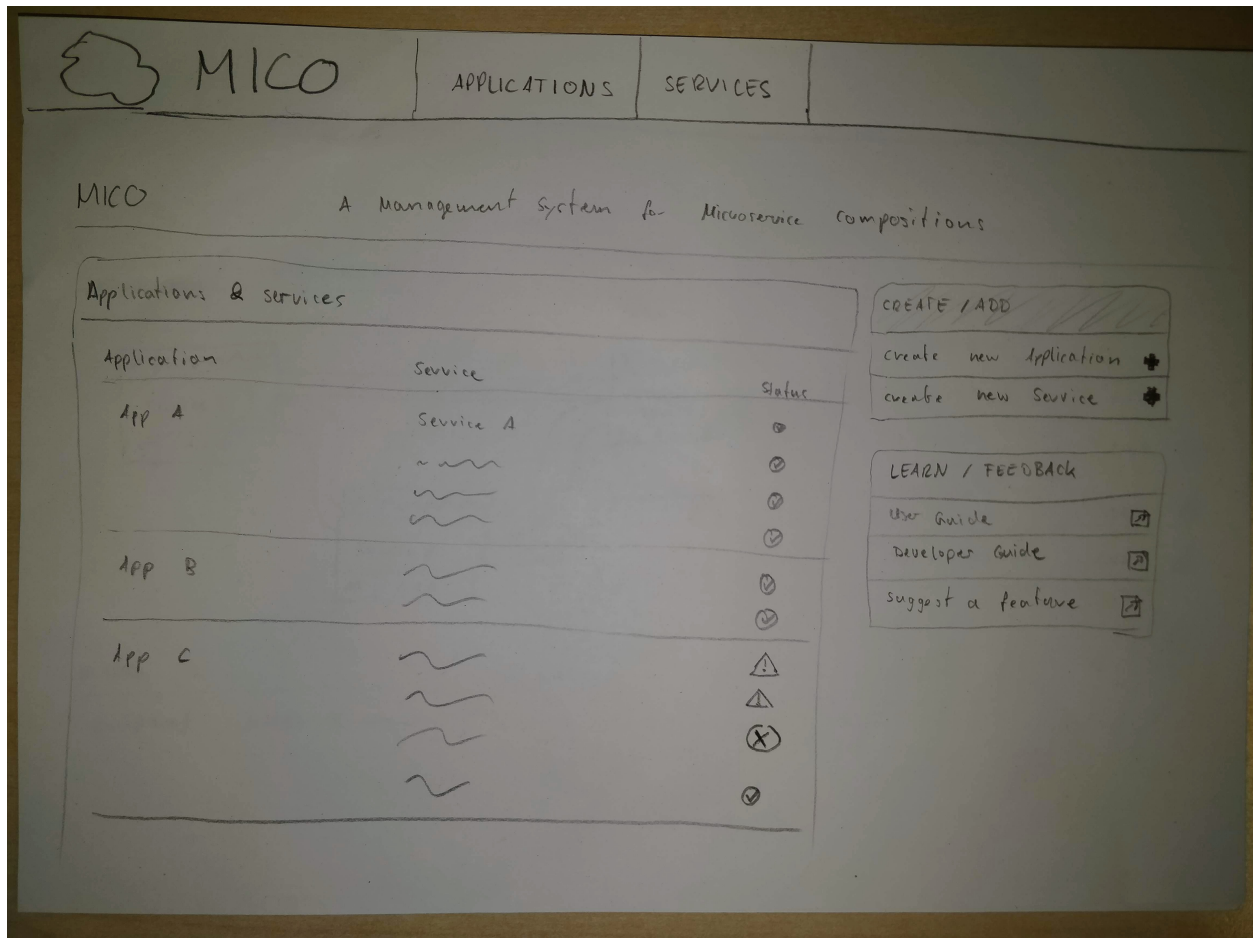


Fig. 1: Mico Dashboard

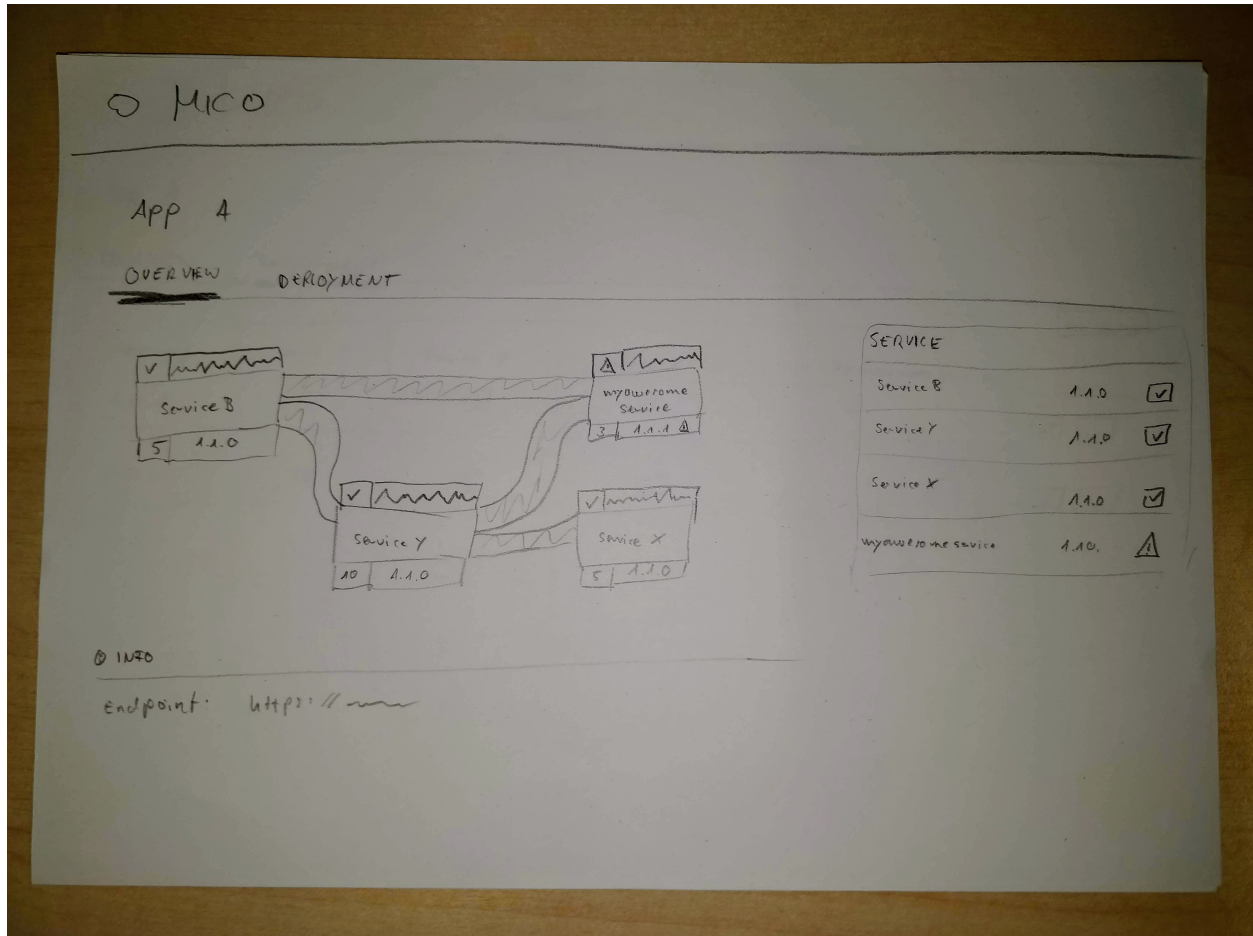


Fig. 2: App-Detail Overview

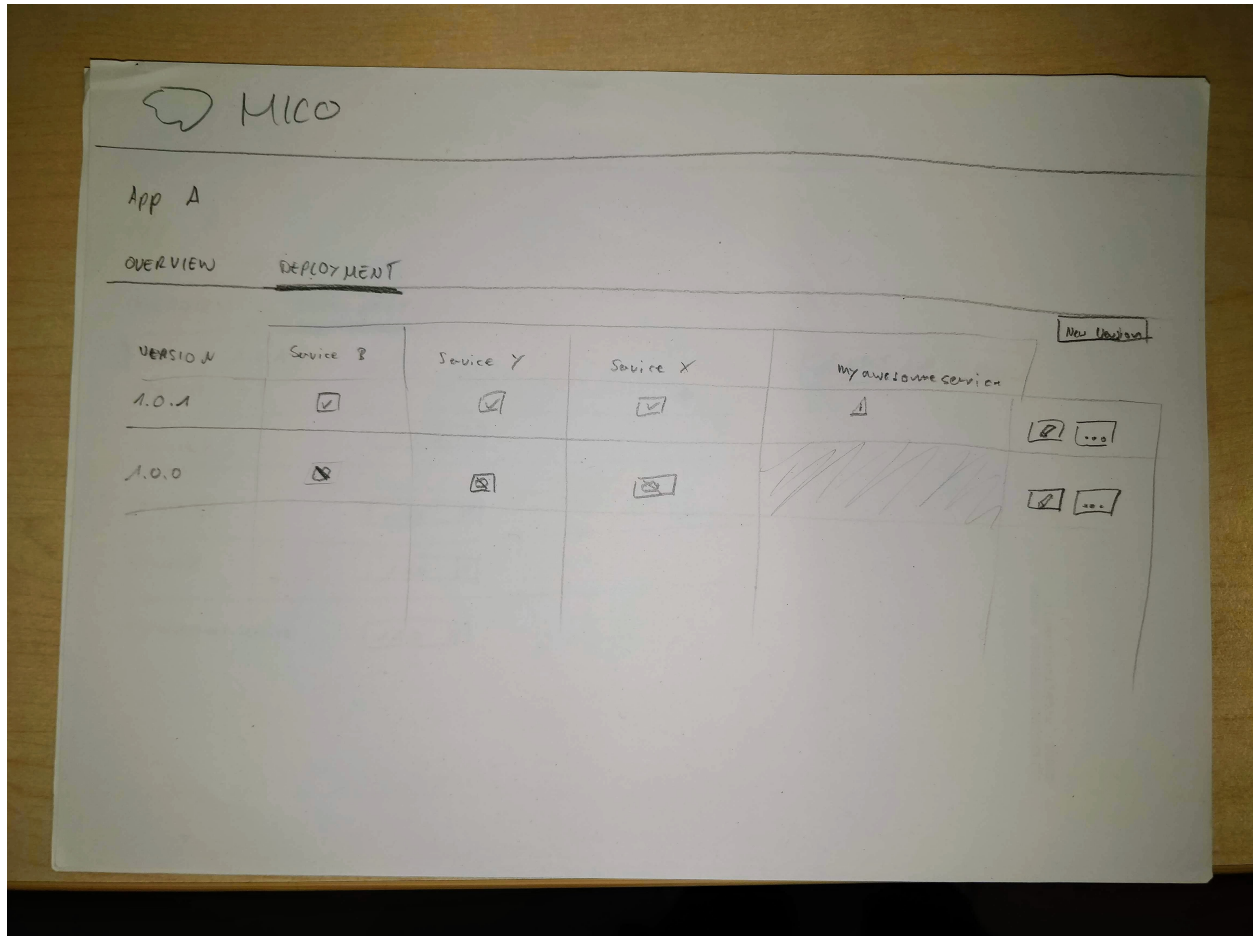


Fig. 3: App-Detail Deploystatus

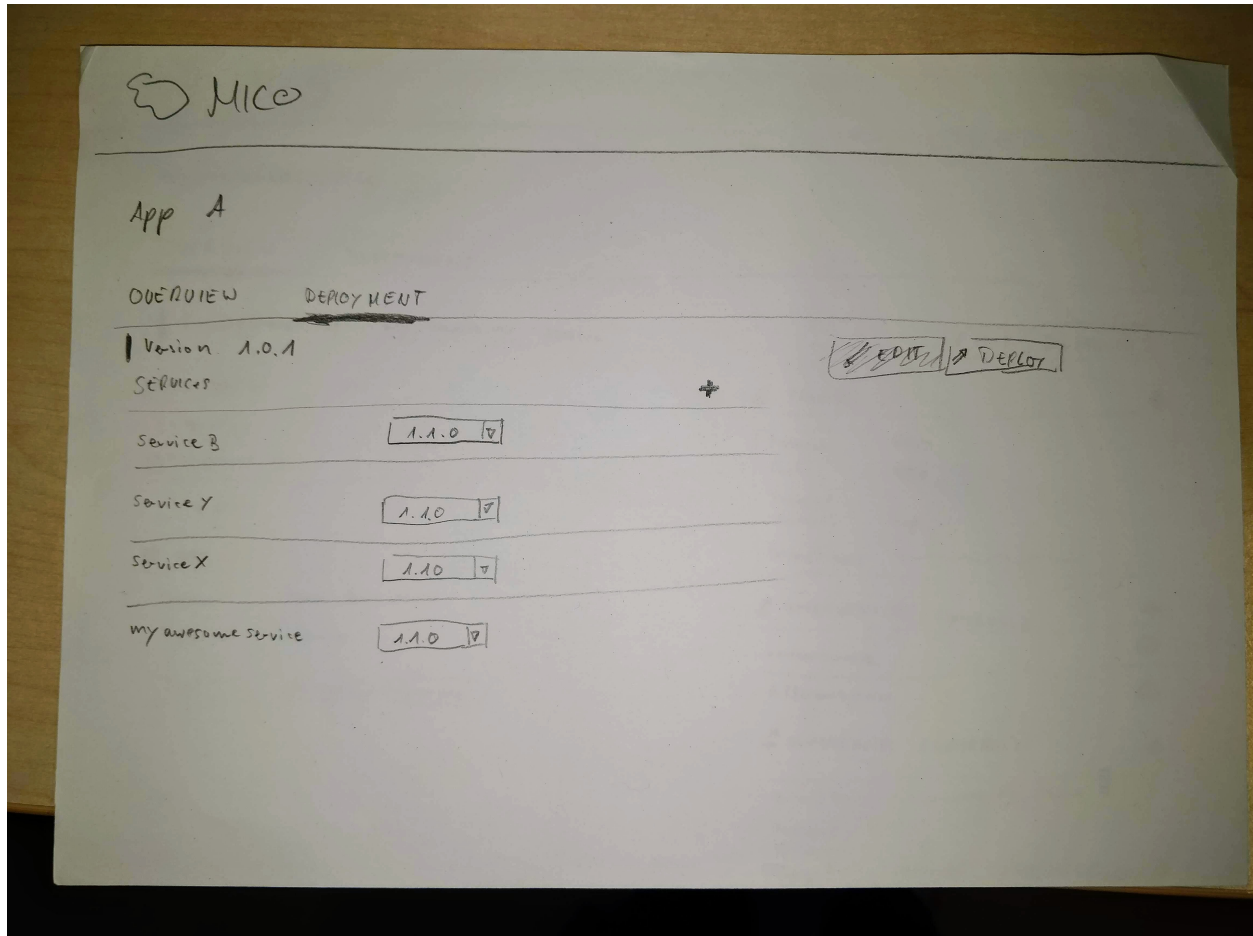


Fig. 4: App-Detail Deploysettings

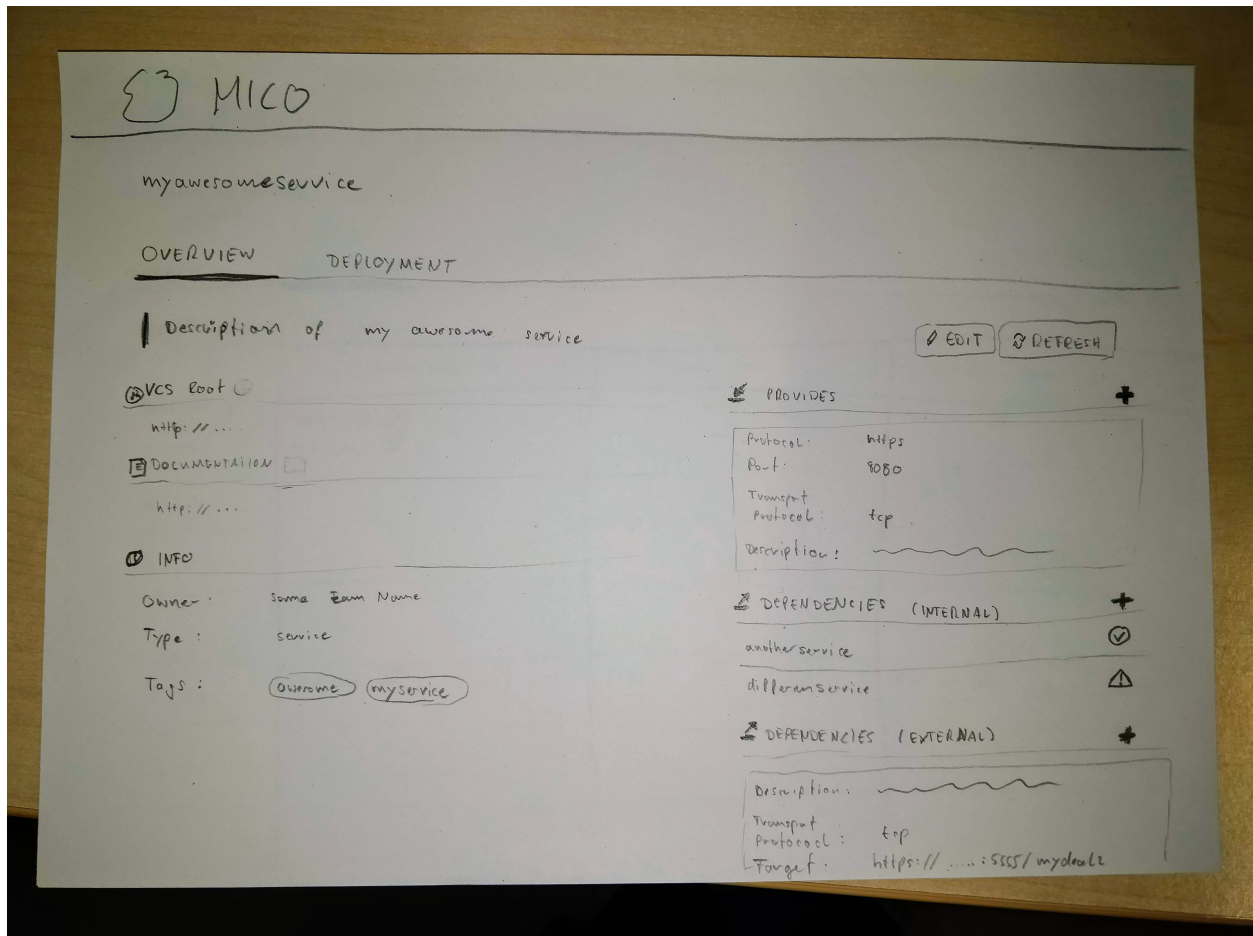



Fig. 5: Service-Detail Overview

 MICO

my awesome service

OVERVIEW DEPLOYMENT

Filter: ☐ Only show deployed versions







| | cluster Node 1 | cluster Node 2 | cluster Node 3 |
|---------|---|---|---|
| VERSION | pod -  <input checked="" type="checkbox"/> | | Pod -  <input checked="" type="checkbox"/> |
| 1.0.10 | pod -  <input checked="" type="checkbox"/> | | |
| 1.0.9 | | | |
| 1.0.8 | | | |
| ... | | | |
| 1.0.20 | Pod -  <input checked="" type="checkbox"/> | pod -  <input checked="" type="checkbox"/> | Pod -  <input checked="" type="checkbox"/> |
| 1.0.19 | | | |
| ... | | | |

Fig. 6: Service-Detail Deploystatus

2.1 MicoApplication

Represents an application represented as a set of instances of *MicoService*

Required fields

- **shortName** A brief name for the application intended for use as a unique identifier.
- **name** The name of the artifact. Intended for humans.
- **version** The version of this application.
- **description** Human readable description of this application.

Optional fields

- **services** The list of included MicoServices
- **serviceDeploymentInfos** The list of service deployment information this application uses for the deployment of the required services. Null values are skipped.
- **contact** Human readable contact information for support purposes.
- **owner** Human readable information for the application owner who is responsible for this application.

2.2 MicoApplicationJobStatus

Represents the job status for a *MicoApplication*. Contains a list of jobs.

Fields

- **applicationShortName** The short name of the application.
- **applicationVersion** The version of the application.
- **status** The aggregated status of jobs for the *MicoApplication*.

- **jobs** The list of jobs for the *MicoApplication*.

2.3 MicoEnvironmentVariable

An environment variable represented as a simple key-value pair. Necessary since Neo4j does not allow to persist properties of composite types.

Fields

- **name** Name of the environment variable.
- **value** Value of the environment variable.

2.4 MicoServiceBackgroundJob

Background job for a *MicoService*. Instances of this class are persisted in the Redis database.

Fields

- **future** The actual job future.
- **serviceShortName** The short name of the corresponding *MicoService*.
- **serviceVersion** The version of the corresponding *MicoService*.
- **type** The type of this job.
- **status** The current status of this Job.
- **errorMessage** An error message in case the job has failed.

2.5 KubernetesDeploymentInfo

Information about the Kubernetes resources that are created through an actual deployment of a *MicoService*.

Optional fields

- **namespace** The namespace in which the Kubernetes deployment is created.
- **deploymentName** The name of the Kubernetes deployment created by a *MicoService*.
- **serviceNames** The names of the Kubernetes services created by *MicoServiceInterface*.

2.6 MicoLabel

Represents a simple key-value pair label. Necessary since Neo4j does not allow to persist Map implementations.

Fields

- **key** Key of the label.
- **value** Value of the label.

2.7 MicoMessage

A simple message associated with a *Type*. Note that this class is only used for business logic purposes and instances are not persisted.

Required fields

- **content** The actual message content.
- **type** The *Type* of this message.

Methods

- **info(String content)** Creates a new *MicoMessage* instance with the type `Type#INFO` and the given message content.
- **error(String content)** Creates a new *MicoMessage* instance with the type `Type#ERROR` and the given message content.
- **warning(String content)** Creates a new *MicoMessage* instance with the type `Type#WARNING` and the given message content.

2.7.1 Type

Enumeration for all types of a *MicoMessage*.

- INFO
- WARNING
- ERROR

2.8 MicoService

Represents a service in the context of MICO.

Required fields

- **shortName** A brief name for the service. In conjunction with the version it must be unique. Pattern is the same as the one for Kubernetes Service names.
- **name** The name of the artifact. Intended for humans. Required only for the usage in the UI.
- **version** The version of this service. E.g. the GitHub release tag.
- **description** Human readable description of this service. Is allowed to be empty (default). Null values are skipped.
- **serviceCrawlingOrigin** Indicates where this service originates from, e.g., GitHub (downloaded and built by MICO) or DockerHub (ready-to-use image). Null is ignored.

Optional fields

- **kafkaEnabled** Indicates whether this service wants to communicate with Kafka. If so this service is handled differently (e.g. it's not mandatory to have interfaces).
- **serviceInterfaces** The list of interfaces this service provides. Is read only. Use special API for updating.
- **dependencies** The list of services that this service requires in order to run normally. Is read only. Use special API for updating.

- **contact** Human readable contact information for support purposes.
- **owner** Human readable information for the service owner who is responsible for this service.
- **dockerfilePath** The relative (to vcsRoot) path to the Dockerfile.
- **dockerImageUri** The fully qualified URI to the image on DockerHub. Either set after the image has been built by MICO (if the service originates from GitHub) or set by the user directly.

2.9 MicoServiceDependency

Represents a dependency of a *MicoService*.

Required fields

- **service** This is the *MicoService* that requires (depends on) the depended service.
- **dependedService** This is the *MicoService* depended by this service.

2.10 MicoServiceDeploymentInfo

Represents the information necessary for deploying a single service.

Required fields

- **service** The *MicoService* this deployment refers to.
- **instanceId** The instanceId of this deployment. It is used to be able to have multiple independent deployments of the same MICO service. Especially for KafkaFaasConnectors this is a must have.

Optional fields

- **replicas** Number of desired instances. Default is 1.
- **labels** Those labels are key-value pairs that are attached to the deployment of this service. Intended to be used to specify identifying attributes that are meaningful and relevant to users, but do not directly imply semantics to the core system. Labels can be used to organize and to select subsets of objects. Labels can be attached to objects at creation time and subsequently added and modified at any time. Each key must be unique for a given object.
- **enviromentVariables** Enviroment variables as key-value pairs that are attached to the deployment of this *MicoService*. These enviroment values can be used by the deployed Micoservice during runtime. This could be useful to pass information to the MicoService that is not known during design time or is likely to change.
- **interfaceConnections** Interface connections includes all required information to be able to connect a *MicoService* with *MicoServiceInterface* of other MicoServices. The backend uses the information to set enviroment variables so that e.g. the frontend knows how to connect to the backend
- **topics** The list of topics that are used in the deployment of this MicoService
- **imagePullPolicy** Indicates whether and when to pull the image. Default is Always.
- **kubernetesDeploymentInfo** Information about the actual Kubernetes resource created by a deploy. Contains details about the used Kubernetes and Services.

2.11 MicoServiceInterface

Represents a interface, e.g., REST API, of a *MicoService*.

Required fields

- **serviceInterfaceName** The name of this *MicoServiceInterface*. Pattern is the same than for Kubernetes Service names.
- **ports** The list of ports. Must not be empty.

Optional fields

- **description** Human readable description of this service interface, e.g., the functionality provided.
- **protocol** The protocol of this interface (e.g. HTTP).

2.12 MicoServicePort

Represents a basic port with a port number and port type (protocol).

Required fields

- **port** The port number of the externally exposed port.
- **type** The type (protocol) of the port. Default port type is MicoPortType.TCP.
- **targetPort** The port inside the container.

2.13 MicoPortType

Enumeration for all port types, e.g., TCP, supported by MICO.

- **TCP** Transmission Control Protocol.
- **UDP** User Datagram Protocol.

2.14 MicoServiceCrawlingOrigin

Enumeration for the various places a service may originate from.

- **GITHUB** Indicates that a service originates from some GitHub respository.
- **DOCKER** Indicates that a service originates from Docker.
- **NOT_DEFINED** Undefined.

2.15 MicoVersion

Wrapper for a version that adds the functionality for a version prefix, so that versions like, e.g., ‘v1.2.3’ are possible.

- **prefix** String prefix of this version, e.g., ‘v’.
- **version** The actual semantic version.

- **valueOf(String version)** Creates a new instance of `MicoVersion` as a result of parsing the specified version string. Prefixes are possible as everything before the first digit in the given version string is treated as a prefix to the actual semantic version. Note that the prefix can only consist of letters.
- **forIntegers(int major, int minor, int patch)** Creates a new instance of `MicoVersion` for the specified version numbers.
- **forIntegersWithPrefix(String prefix, int major, int minor, int patch)** Creates a new instance of `MicoVersion` for the specified version numbers with the specified prefix string.

2.16 MicoInterfaceConnection

An interface connection contains the the information needed to connect a *MicoService* to an *MicoServiceInterface* of another *MicoService*. Instances of this class are persisted as nodes in the Neo4j database.

Required fields

- **environmentVariableName** Name of the environment variable that is used to set the fully qualified name of an interface.
- **micoServiceInterfaceName** Name of the *MicoServiceInterface* of an *MicoService*.
- **micoServiceShortName** Name of the *MicoService*.

2.17 MicoTopic

A Topic represented a kafka-topic. Instances of this class are persisted as nodes in the neo4j database.

Required Fields

- **name** Name of the topic

2.18 MicoTopicRole

Represents a role of a *MicoTopic*. An instance of this class is persisted as a relationship between a `MicoServiceDeploymentInfo` and a `MicoTopic` node in the neo4j database.

Required Fields

- **serviceDeploymentInfos** This is the `MicoServiceDeploymentInfo` that includes the `MicoTopicRole`
- **topic** This is the `MicoTopic` included by the `MicoTopicRole#serviceDeploymentInfos`
- **role** This is the role of the `MicoTopicRole`

2.18.1 Role

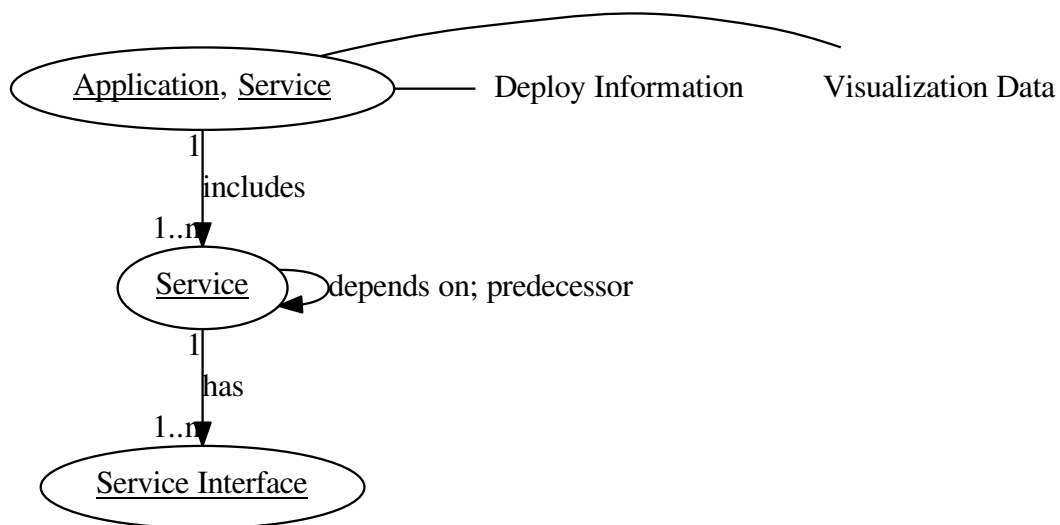
Enumeration for all topic roles

- INPUT
- OUTPUT
- DEAD_LETTER
- INVALID_MESSAGE

- TEST_MESSAGE_OUTPUT

3.1 Overview

Initial graph based on *Database Technology (Relational/Graph)*.



3.2 Service

Attributes describing a *MICO Service* are mainly taken from *Pivio*.

| Attribute | Description | Example |
|-----------------|--|---|
| ID* | UUID (Unique for every version) | 7e7e1828-e845-11e8-9f32-f2801f1b9fd1 |
| shortName* | Unique short name (java Package Style) | de.uni-stuttgart.mico. hello-world |
| name* | Display name | Hello World |
| description* | Human readable description of the service | A simple Hello World Service. |
| version* | Semantic version version number, commit hash or revision | 1.0.2, 1.2.0a, d99ada433bb1d69571... |
| vcsroot | Url to vcs root (e.g. GitHub link with branch) | https://github.com/UST-MICO/ mico |
| dockerfile | Path to dockerfile relative to vcsroot | mico-core/Dockerfile |
| contact | Human readable contact information. | Max Musterman, musterman@muster.de |
| tags | To be defined | |
| links | To be defined | |
| owner | Human readable team information. | Max Musterman, Company Y |
| externalService | A boolean which indicates if a service is external | true |

3.3 DependsOn

DependsOn is an edge between two services. A service depends on another service. Additionally, it provides the attribute version. This attribute allows to implement an auto-update feature. It contains a version string in the [semantic version](#) format. This allows to determine the most recent compatible service.

3.4 Service Interface

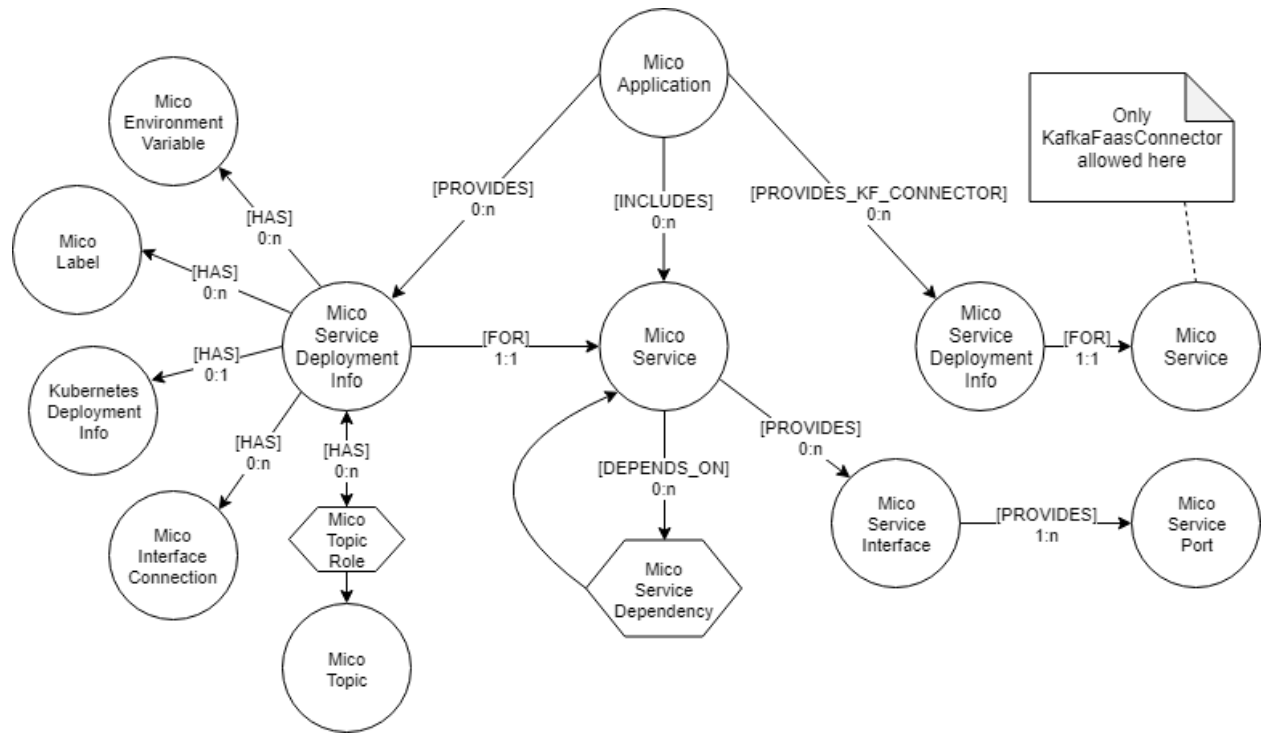
| Attribute | Description | Example |
|-----------------------|---|-----------------------------|
| serviceInterfaceName* | Unique short name suffix for shortName of service. | rest |
| description* | Human readable description | This is a service interface |
| port* | This is a port number or a variable name | 1024 |
| protocol* | The name of the protocol which is supported by this interface | gRPC, SQL, CYPHER |
| transport_protocol* | This protocol is used to transport the previous protocol | http, mqtt |
| public_dns* | To be defined | |

3.5 Application

Because an application is a service itself it also includes all attributes of a service even when not explicitly stated in the table.

| Attribute | Description | Example |
|-----------|---------------------------------|--------------------------------------|
| ID* | UUID (Unique for every version) | 7e7e1828-e845-11e8-9f32-f2801f1b9fd1 |

3.6 Possible Nodes and Relationships



This section includes information about M3 - the messaging focus of MICO.

4.1 CloudEvents

[CloudEvents](#) is a specification for describing event data in a common way. We want to support this vendor-neutral specification.

Currently we depend on the [CloudEvents specification v0.2](#).

4.1.1 Attributes

Context Attributes (REQUIRED):

- **type** – cloud event definition
- **specversion** – cloud event definition
- **source** – cloud event definition
- **id** – change id if message was changed significantly (data translation, ignoring addition only changes) if id changed add previous message id as **createdFrom** to message (overwriting previous **createdFrom**)

Context Attributes (OPTIONAL):

- **time** – cloud event definition Change to current time if id changed
- **schemauri** – cloud event definition
- **contenttype** – cloud event definition, contenttype + schemauri define the complete message format
- **subject** - definition of [CloudEvents spec v0.3](#): Identifying the subject of the event in context metadata (opposed to only in the data payload) is particularly helpful in generic subscription filtering scenarios where middleware is unable to interpret the data content. In the above example, the subscriber might only be interested in blobs with names ending with ‘.jpg’ or ‘.jpeg’ and the subject attribute allows for constructing a simple and efficient string-suffix filter for that subset of events.

- **createdfrom** – id of the message this message was directly created from
- **correlationid** – always keep if set, even in answers to this message (often the id of first message in a createdfrom chain)
- **route** – list of topics and functions the message was sent to (updated by sender), keep until destination of message The entries are json objects with a `type`, `id` and `timestamp` attribute. The `type` is either "topic" or "faas-function". The `id` is either the topic name or the function id. The `timestamp` is the time when the entry was added (before sending the message to the topic or function).
- **routingstip** – stack of topics the message should be routed to (last entry is the next destination) The stack is represented as a json list. The entries are lists of topic names (`["topic1", "topic2"]`). The message is routed to ALL topics in the same list simultaneously.
- **istestmessage** – boolean flag for test messages
- **filteroutbeforetopic** – only applies to test messages. Message should never be sent to specified topic but instead be routed to the collection topic for test messages to be inspected by a user.
- **iserrormessage** – boolean flag for error messages The message that caused this error should be added as data in the error message. The id of the message causing the error should be used as the `createdfrom` value in the error message.
- **errormessage** – string containing the (short) error message
- **errortrace** – string containing additional information about the error. This could be a full stacktrace, part of a log or even json containing structured information about the error.
- **expirydate** – for messages that should be considered stale after this date. Stale messages should not be consumed. They can be routed to the error message collection topic or a special topic for stale messages only.
- **sequenceid** – message id for messages split into a sequence of messages
- **sequencenumber** – the position of this message in the sequence as an Integer. The sequence MUST start with a value of 1 increase by 1 for each subsequent value.
- **sequencesize** – the number of messages in the sequence as an Integer (only if known, this attribute is not required)
- **returntopic** – the return address for replies to this message (implementation details to be decided by the user)
- **dataref** – [cloud event v0.3 extension](#)
- **partitionkey** – used as key when sending a kafka message. [cloud event v0.3 extension](#)

Data Attribute:

- **data** – cloud event definition

4.1.2 Examples

Minimal:

```
{
  "specversion" : "0.2",
  "type" : "io.github.ust.mico.result",
  "source" : "/router",
  "id" : "A234-1234-1234",
  "time" : "2019-05-08T17:31:00Z",
  "contenttype" : "application/json",
  "data" : {
    "key" : "value"
  }
}
```

(continues on next page)

(continued from previous page)

```
}
}
```

With routing fields:

```
{
  "specversion" : "0.2",
  "type" : "io.github.ust.mico.result",
  "source" : "/router",
  "id" : "A234-1234-1234",
  "correlationid": "A234-1234-1234",
  "time" : "2019-05-08T17:31:00Z",
  "contenttype" : "application/json",
  "data" : {
    "key" : "value"
  },
  "route": [
    {
      "type": "topic",
      "id": "first-topic",
      "timestamp": "2019-05-08T17:31:30Z"
    },
    {
      "type": "faas-function",
      "id": "xmlToJson",
      "timestamp": "2019-05-08T17:31:55Z"
    }
  ],
  "routingslip": [
    ["final-destination"],
    ["filter", "debug-topic"],
    ["next-destination"]
  ]
}
```

4.2 Error Report Messages

As described by the EAI Patterns [Invalid Message Channel](#) and [Dead Letter Channel](#) we need a way to handle messages which can not be processed and or routed. We adopt the concept of the Pattern and use topics to realize them. Messages which can not be processed are reported to the Invalid Message Topic and messages which can not be routed are reported to the Dead Letter Topic. The messages in these topics use a CloudEvent envelope and the following error report message format in the data field of the envelope.

4.2.1 Attributes

Used attributes in the CloudEvent envelope:

- **createdfrom** – id of the message which triggered the error

Error report Message Attributes:

- **errorMessage** – human-readable error message.
- **inputTopic** – the original message which triggered the error was read from this topic

- **outputTopic** – the standard output topic. FaaS functions can overwrite this
- **faasGateway** - the gateway of the configured FaaS function
- **faasFunctionName** - the name of the FaaS function
- **reportingComponentName** – the name of the component which reports the error

5.1 Structure Overview

This is how the structure of the API should look like.

- *GET /services*
 - *GET /(shortName)/*
 - * *GET /(version)/*
 - *GET /interfaces/*
 - *GET /(serviceInterfaceName)/publicIP/*
 - *GET /dependees/*
 - *GET /dependers/*
 - *GET /dependencyGraph/*
 - *POST /promote/*
 - *GET /status/*
 - *GET /import/*
 - * *GET /github/*
 - * *POST /github/*
- *GET /applications*
 - *GET /(shortName)/*
 - * *GET /(version)/*
 - *GET /services/*
 - *GET /deploymentInformation/*
 - *POST /promote/*

- `GET /status/`
- `POST /deploy/`
- `GET /jobs/`
 - `GET /(id)/`

5.2 Openapi Specification

Current API status:

**GET /actuator
links**

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

**GET /actuator/configprops
handle**

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

**GET /actuator/health
handle**

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

**GET /actuator/health/{component}
handle**

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

**GET /actuator/health/{component}/{instance}
handle**

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /actuator/info
handle

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /actuator/loggers
handle

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /actuator/loggers/{name}
handle

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

POST /actuator/loggers/{name}
handle

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /actuator/metrics
handle

Status Codes

- 200 OK – OK

- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /actuator/metrics/{requiredMetricName}
handle

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /applications
getAllApplications

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- `_embedded[]._links[].href` (*string*) –
- `_embedded[]._links[].templated` (*boolean*) –
- `_embedded[].contact` (*string*) – Human readable contact information for support purposes.
- `_embedded[].deploymentStatus` (*string*) – Holds the current deployment status of this application. (read only)
- `_embedded[].description` (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- `_embedded[].name` (*string*) – Human readable name of the application. (required)
- `_embedded[].owner` (*string*) – Human readable information for the application owner, who is responsible for this application.
- `_embedded[].services[].contact` (*string*) – Human readable contact information for support purposes.
- `_embedded[].services[].description` (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- `_embedded[].services[].dockerImageUri` (*string*) – The fully qualified URI to the image on DockerHub.
- `_embedded[].services[].dockerfilePath` (*string*) – The relative path to the Dockerfile.

- **_embedded[] .services[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **_embedded[] .services[] .name** (*string*) – A human readable name of the MicroService. (required)
- **_embedded[] .services[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **_embedded[] .services[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **_embedded[] .services[] .shortName** (*string*) – A unique name of the MicroService. (required)
- **_embedded[] .services[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **_embedded[] .shortName** (*string*) – Unique short name of the application. (required)
- **_embedded[] .version** (*string*) – The version of this application. (required)
- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –

POST /applications createApplication

Request JSON Object

- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **name** (*string*) – Human readable name of the application. (required)
- **owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **shortName** (*string*) – Unique short name of the application. (required)
- **version** (*string*) – The version of this application. (required)

Status Codes

- **200 OK** – OK
- **201 Created** – Created
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **deploymentStatus** (*string*) – Holds the current deployment status of this application. (read only)

- **description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **name** (*string*) – Human readable name of the application. (required)
- **owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **services[] .contact** (*string*) – Human readable contact information for support purposes.
- **services[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **services[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **services[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **services[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **services[] .name** (*string*) – A human readable name of the MicoService. (required)
- **services[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **services[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **services[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **services[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **shortName** (*string*) – Unique short name of the application. (required)
- **version** (*string*) – The version of this application. (required)

GET /applications/{shortName}
getApplicationsByShortName

Parameters

- **shortName** (*string*) – shortName

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_embedded[] ._links[] .href** (*string*) –
- **_embedded[] ._links[] .templated** (*boolean*) –
- **_embedded[] .contact** (*string*) – Human readable contact information for support purposes.

- **_embedded[].deploymentStatus** (*string*) – Holds the current deployment status of this application. (read only)
- **_embedded[].description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **_embedded[].name** (*string*) – Human readable name of the application. (required)
- **_embedded[].owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **_embedded[].services[].contact** (*string*) – Human readable contact information for support purposes.
- **_embedded[].services[].description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **_embedded[].services[].dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **_embedded[].services[].dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **_embedded[].services[].gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **_embedded[].services[].name** (*string*) – A human readable name of the MicroService. (required)
- **_embedded[].services[].owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **_embedded[].services[].serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **_embedded[].services[].shortName** (*string*) – A unique name of the MicroService. (required)
- **_embedded[].services[].version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **_embedded[].shortName** (*string*) – Unique short name of the application. (required)
- **_embedded[].version** (*string*) – The version of this application. (required)
- **_links[].href** (*string*) –
- **_links[].templated** (*boolean*) –

DELETE /applications/{shortName}
deleteAllVersionsOfAnApplication

Parameters

- **shortName** (*string*) – shortName

Status Codes

- **200 OK** – OK
- **204 No Content** – No Content
- **401 Unauthorized** – Unauthorized

- [403 Forbidden](#) – Forbidden

GET /applications/{shortName}/{version}
getApplicationByShortNameAndVersion

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- [200 OK](#) – OK
- [401 Unauthorized](#) – Unauthorized
- [403 Forbidden](#) – Forbidden
- [404 Not Found](#) – Not Found

Response JSON Object

- **_links[]**.**href** (*string*) –
- **_links[]**.**templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **deploymentStatus** (*string*) – Holds the current deployment status of this application. (read only)
- **description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **name** (*string*) – Human readable name of the application. (required)
- **owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **services[]**.**contact** (*string*) – Human readable contact information for support purposes.
- **services[]**.**description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **services[]**.**dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **services[]**.**dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **services[]**.**gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **services[]**.**name** (*string*) – A human readable name of the MicoService. (required)
- **services[]**.**owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **services[]**.**serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **services[]**.**shortName** (*string*) – A unique name of the MicoService. (required)

- **services[]**.**version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **shortName** (*string*) – Unique short name of the application. (required)
- **version** (*string*) – The version of this application. (required)

PUT /applications/{shortName}/{version}
updateApplication

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **name** (*string*) – Human readable name of the application. (required)
- **owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **shortName** (*string*) – Unique short name of the application. (required)
- **version** (*string*) – The version of this application. (required)

Status Codes

- **200 OK** – OK
- **201 Created** – Created
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_links[]**.**href** (*string*) –
- **_links[]**.**templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **deploymentStatus** (*string*) – Holds the current deployment status of this application. (read only)
- **description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **name** (*string*) – Human readable name of the application. (required)
- **owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **services[]**.**contact** (*string*) – Human readable contact information for support purposes.
- **services[]**.**description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)

- **services[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **services[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **services[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **services[] .name** (*string*) – A human readable name of the MicoService. (required)
- **services[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **services[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **services[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **services[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **shortName** (*string*) – Unique short name of the application. (required)
- **version** (*string*) – The version of this application. (required)

DELETE /applications/{shortName}/{version}
deleteApplication

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

POST /applications/{shortName}/{version}/deploy
deploy

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- `_links[].href(string)` –
- `_links[].templated(boolean)` –
- `jobs[].errorMessage(string)` – An error message in case the job has failed (read only)
- `jobs[].id(string)` – The generated job id. (read only)
- `jobs[].serviceShortName(string)` – The name of the corresponding service. (read only)
- `jobs[].serviceVersion(string)` – The version of the corresponding service. (read only)
- `jobs[].status(string)` – The current status of the job. (read only)
- `jobs[].type(string)` – The type of job. (read only)
- `status(string)` – The aggregated status of jobs for an application. (read only)

GET /applications/{shortName}/{version}/deploymentInformation/{serviceShortName}
getServiceDeploymentInformation

Parameters

- `serviceShortName(string)` – serviceShortName
- `shortName(string)` – shortName
- `version(string)` – version

Status Codes

- [200 OK](#) – OK
- [401 Unauthorized](#) – Unauthorized
- [403 Forbidden](#) – Forbidden
- [404 Not Found](#) – Not Found

Response JSON Object

- `_links[].href(string)` –
- `_links[].templated(boolean)` –
- `environmentVariables[].name(string)` – Name of the environment variable. (required)
- `environmentVariables[].value(string)` – Value of the environment variable.
- `imagePullPolicy(string)` – Indicates whether and when to pull the image. Null is ignored.
- `labels[].key(string)` – Key of the label. (required)
- `labels[].value(string)` – Value of the label.
- `replicas(integer)` – Number of desired instances. Defaults to 1.
- `restartPolicy(string)` – Restart policy for all containers. Null is ignored.

PUT /applications/{shortName}/{version}/deploymentInformation/{serviceShortName}
updateServiceDeploymentInformation

Parameters

- **serviceShortName** (*string*) – serviceShortName
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **environmentVariables[] .name** (*string*) – Name of the environment variable. (required)
- **environmentVariables[] .value** (*string*) – Value of the environment variable.
- **imagePullPolicy** (*string*) – Indicates whether and when to pull the image. Null is ignored.
- **labels[] .key** (*string*) – Key of the label. (required)
- **labels[] .value** (*string*) – Value of the label.
- **replicas** (*integer*) – Number of desired instances. Defaults to 1.
- **restartPolicy** (*string*) – Restart policy for all containers. Null is ignored.

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **environmentVariables[] .name** (*string*) – Name of the environment variable. (required)
- **environmentVariables[] .value** (*string*) – Value of the environment variable.
- **imagePullPolicy** (*string*) – Indicates whether and when to pull the image. Null is ignored.
- **labels[] .key** (*string*) – Key of the label. (required)
- **labels[] .value** (*string*) – Value of the label.
- **replicas** (*integer*) – Number of desired instances. Defaults to 1.
- **restartPolicy** (*string*) – Restart policy for all containers. Null is ignored.

**POST /applications/{shortName}/{version}/promote
promoteApplication**

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **version** (*string*) – The version (in semantic version format). (required)

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **deploymentStatus** (*string*) – Holds the current deployment status of this application. (read only)
- **description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **name** (*string*) – Human readable name of the application. (required)
- **owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **services[] .contact** (*string*) – Human readable contact information for support purposes.
- **services[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **services[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **services[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **services[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **services[] .name** (*string*) – A human readable name of the MicoService. (required)
- **services[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **services[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **services[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **services[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **shortName** (*string*) – Unique short name of the application. (required)
- **version** (*string*) – The version of this application. (required)

GET /applications/{shortName}/{version}/services
getServicesFromApplication

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_embedded[]._links[] .href** (*string*) –
- **_embedded[]._links[] .templated** (*boolean*) –
- **_embedded[] .contact** (*string*) – Human readable contact information for support purposes.
- **_embedded[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **_embedded[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **_embedded[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **_embedded[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **_embedded[] .name** (*string*) – A human readable name of the MicoService. (required)
- **_embedded[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **_embedded[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **_embedded[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **_embedded[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –

DELETE /applications/{shortName}/{version}/services/{serviceShortName}
deleteServiceFromApplication

Parameters

- **serviceShortName** (*string*) – serviceShortName
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

POST /applications/{shortName}/{version}/services/{serviceShortName}/{serviceVersion}
addServiceToApplication

Parameters

- **serviceShortName** (*string*) – serviceShortName
- **serviceVersion** (*string*) – serviceVersion
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

GET /applications/{shortName}/{version}/status
getStatusOfApplication

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[]**.href (*string*) –
- **_links[]**.templated (*boolean*) –
- **serviceStatuses[]**.applicationsUsingThisService[].contact (*string*) – Human readable contact information for support purposes.
- **serviceStatuses[]**.applicationsUsingThisService[].deploymentStatus (*string*) – Holds the current deployment status of this application. (read only)

- **serviceStatuses[] .applicationsUsingThisService[] .description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **serviceStatuses[] .applicationsUsingThisService[] .name** (*string*) – Human readable name of the application. (required)
- **serviceStatuses[] .applicationsUsingThisService[] .owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **serviceStatuses[] .applicationsUsingThisService[] .shortName** (*string*) – Unique short name of the application. (required)
- **serviceStatuses[] .applicationsUsingThisService[] .version** (*string*) – The version of this application. (required)
- **serviceStatuses[] .availableReplicas** (*integer*) – Counter for the number of replicas of the corresponding MicoService that are actually available.
- **serviceStatuses[] .errorMessages[]** (*string*) –
- **serviceStatuses[] .interfacesInformation[] .externalIps[]** (*string*) –
- **serviceStatuses[] .interfacesInformation[] .name** (*string*) – Name of the MicoServiceInterface.
- **serviceStatuses[] .name** (*string*) – Name of the MicoService.
- **serviceStatuses[] .nodeMetrics[] .averageCpuLoad** (*integer*) – The average CPU load of all pods of one MicoService running on this Node.
- **serviceStatuses[] .nodeMetrics[] .averageMemoryUsage** (*integer*) – The average memory usage of all pods of one MicoService running on this Node.
- **serviceStatuses[] .nodeMetrics[] .nodeName** (*string*) – Name of the Kubernetes Node.
- **serviceStatuses[] .podsInformation[] .hostIp** (*string*) – IP address of the host to which the pod is assigned.
- **serviceStatuses[] .podsInformation[] .metrics .available** (*boolean*) – Indicates if a pod is available.
- **serviceStatuses[] .podsInformation[] .metrics .cpuLoad** (*integer*) – CPU load of a pod.
- **serviceStatuses[] .podsInformation[] .metrics .memoryUsage** (*integer*) – Memory usage of a pod.
- **serviceStatuses[] .podsInformation[] .nodeName** (*string*) – Name of the node the pod is running on.
- **serviceStatuses[] .podsInformation[] .phase** (*string*) – States the lifecycle of a pod: Pending, Running, Succeeded, Failed, Unknown, Completed, CrashLoopBackOff
- **serviceStatuses[] .podsInformation[] .podName** (*string*) – Name of the Pod.
- **serviceStatuses[] .podsInformation[] .restarts** (*integer*) – Counter for restarts of all containers of a pod.

- **serviceStatuses[] .podsInformation[] .startTime** (*string*) – The point in time the pod has been started.
- **serviceStatuses[] .requestedReplicas** (*integer*) – Counter for the number of replicas of the corresponding MicoService that should be available.
- **serviceStatuses[] .shortName** (*string*) – Short name of the MicoService.
- **serviceStatuses[] .version** (*string*) – Version of the MicoService.
- **totalNumberOfAvailableReplicas** (*integer*) – Number of replicas of all services that are available in a MicoApplication.
- **totalNumberOfMicoServices** (*integer*) – Number of MicoServices of a MicoApplication.
- **totalNumberOfPods** (*integer*) – Number of pods created by all MicoServices in a MicoApplication.
- **totalNumberOfRequestedReplicas** (*integer*) – Number of replicas of all services that should be available in a MicoApplication.

GET /error
errorHtml

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –
- **viewName** (*string*) –

HEAD /error
errorHtml

Status Codes

- **200 OK** – OK
- **204 No Content** – No Content
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –

- **viewName** (*string*) –

POST /error
errorHtml

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –
- **viewName** (*string*) –

PUT /error
errorHtml

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –
- **viewName** (*string*) –

DELETE /error
errorHtml

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –
- **viewName** (*string*) –

OPTIONS /error
errorHtml

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –
- **viewName** (*string*) –

PATCH /error
errorHtml

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

Response JSON Object

- **empty** (*boolean*) –
- **reference** (*boolean*) –
- **status** (*string*) –
- **view.contentType** (*string*) –
- **viewName** (*string*) –

GET /jobs
getAllJobs

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

- [404 Not Found](#) – Not Found

Response JSON Object

- **`__embedded[].__links[].__href`** (*string*) –
- **`__embedded[].__links[].__templated`** (*boolean*) –
- **`__embedded[].__errorMessage`** (*string*) – An error message in case the job has failed (read only)
- **`__embedded[].__id`** (*string*) – The generated job id. (read only)
- **`__embedded[].__serviceShortName`** (*string*) – The name of the corresponding service. (read only)
- **`__embedded[].__serviceVersion`** (*string*) – The version of the corresponding service. (read only)
- **`__embedded[].__status`** (*string*) – The current status of the job. (read only)
- **`__embedded[].__type`** (*string*) – The type of job. (read only)
- **`__links[].__href`** (*string*) –
- **`__links[].__templated`** (*boolean*) –

GET /jobs/{id}
getJobById

Parameters

- **`id`** (*string*) – id

Status Codes

- [200 OK](#) – OK
- [401 Unauthorized](#) – Unauthorized
- [403 Forbidden](#) – Forbidden
- [404 Not Found](#) – Not Found

Response JSON Object

- **`__links[].__href`** (*string*) –
- **`__links[].__templated`** (*boolean*) –
- **`errorMessage`** (*string*) – An error message in case the job has failed (read only)
- **`id`** (*string*) – The generated job id. (read only)
- **`serviceShortName`** (*string*) – The name of the corresponding service. (read only)
- **`serviceVersion`** (*string*) – The version of the corresponding service. (read only)
- **`status`** (*string*) – The current status of the job. (read only)
- **`type`** (*string*) – The type of job. (read only)

DELETE /jobs/{id}
deleteJob

Parameters

- **`id`** (*string*) – id

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

GET /jobs/{shortName}/{version}/status
getJobStatusByApplicationShortNameAndVersion

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **jobs[] .errorMessage** (*string*) – An error message in case the job has failed (read only)
- **jobs[] .id** (*string*) – The generated job id. (read only)
- **jobs[] .serviceShortName** (*string*) – The name of the corresponding service. (read only)
- **jobs[] .serviceVersion** (*string*) – The version of the corresponding service. (read only)
- **jobs[] .status** (*string*) – The current status of the job. (read only)
- **jobs[] .type** (*string*) – The type of job. (read only)
- **status** (*string*) – The aggregated status of jobs for an application. (read only)

GET /services
getServiceList

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_embedded[] ._links[] .href** (*string*) –
- **_embedded[] ._links[] .templated** (*boolean*) –

- **_embedded[] .contact** (*string*) – Human readable contact information for support purposes.
- **_embedded[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **_embedded[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **_embedded[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **_embedded[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **_embedded[] .name** (*string*) – A human readable name of the MicoService. (required)
- **_embedded[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **_embedded[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **_embedded[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **_embedded[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –

POST /services
createService

Request JSON Object

- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **name** (*string*) – A human readable name of the MicoService. (required)
- **owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **shortName** (*string*) – A unique name of the MicoService. (required)
- **version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized

- [403 Forbidden](#) – Forbidden
- [404 Not Found](#) – Not Found

Response JSON Object

- `_links[] .href (string)` –
- `_links[] .templated (boolean)` –
- `contact (string)` – Human readable contact information for support purposes.
- `description (string)` – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- `dockerImageUri (string)` – The fully qualified URI to the image on DockerHub.
- `dockerfilePath (string)` – The relative path to the Dockerfile.
- `gitCloneUrl (string)` – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- `name (string)` – A human readable name of the MicoService. (required)
- `owner (string)` – Human readable information for the services owner, who is responsible for this service.
- `serviceCrawlingOrigin (string)` – Indicates where this service originates from. (read only)
- `shortName (string)` – A unique name of the MicoService. (required)
- `version (string)` – The version of this service. Refers to GitHub release tag. (required)

GET /services/import/github
getVersionsFromGitHub

Query Parameters

- `url (string)` – url

Status Codes

- [200 OK](#) – OK
- [401 Unauthorized](#) – Unauthorized
- [403 Forbidden](#) – Forbidden
- [404 Not Found](#) – Not Found

Response JSON Object

- `_embedded[] ._links[] .href (string)` –
- `_embedded[] ._links[] .templated (boolean)` –
- `_embedded[] .version (string)` – The version (in semantic version format). (required)
- `_links[] .href (string)` –
- `_links[] .templated (boolean)` –

POST /services/import/github
importMicoServiceFromGitHub

Request JSON Object

- **url** (*string*) – The url to the remote repository to crawl from. (required)
- **version** (*string*) – The remote release tag. Defaults to ‘latest’.

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[]**.**href** (*string*) –
- **_links[]**.**templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **name** (*string*) – A human readable name of the MicoService. (required)
- **owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **shortName** (*string*) – A unique name of the MicoService. (required)
- **version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

GET /services/{shortName}
getVersionsOfService

Parameters

- **shortName** (*string*) – shortName

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_embedded[]**.**_links[]**.**href** (*string*) –
- **_embedded[]**.**_links[]**.**templated** (*boolean*) –

- **__embedded[] .contact** (*string*) – Human readable contact information for support purposes.
- **__embedded[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **__embedded[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **__embedded[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **__embedded[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **__embedded[] .name** (*string*) – A human readable name of the MicoService. (required)
- **__embedded[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **__embedded[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **__embedded[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **__embedded[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **__links[] .href** (*string*) –
- **__links[] .templated** (*boolean*) –

DELETE /services/{shortName}
deleteAllVersionsOfService

Parameters

- **shortName** (*string*) – shortName

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

GET /services/{shortName}/{version}
getServiceByShortNameAndVersion

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[]**.**href** (*string*) –
- **_links[]**.**templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **name** (*string*) – A human readable name of the MicoService. (required)
- **owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **shortName** (*string*) – A unique name of the MicoService. (required)
- **version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

PUT /services/{shortName}/{version}
updateService

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **name** (*string*) – A human readable name of the MicoService. (required)
- **owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **shortName** (*string*) – A unique name of the MicoService. (required)
- **version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

Status Codes

- **200 OK** – OK

- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[]**.href (*string*) –
- **_links[]**.templated (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.
- **description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **name** (*string*) – A human readable name of the MicoService. (required)
- **owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **shortName** (*string*) – A unique name of the MicoService. (required)
- **version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

DELETE /services/{shortName}/{version}
deleteService

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

GET /services/{shortName}/{version}/dependees
getDependees

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_embedded[]._links[] .href** (*string*) –
- **_embedded[]._links[] .templated** (*boolean*) –
- **_embedded[] .contact** (*string*) – Human readable contact information for support purposes.
- **_embedded[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **_embedded[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **_embedded[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **_embedded[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **_embedded[] .name** (*string*) – A human readable name of the MicoService. (required)
- **_embedded[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **_embedded[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **_embedded[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **_embedded[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –

DELETE /services/{shortName}/{version}/dependees
deleteAllDependees

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

POST /services/{shortName}/{version}/dependees/{dependeeShortName}/{dependeeVersion}
createNewDependee

Parameters

- **dependeeShortName** (*string*) – dependeeShortName
- **dependeeVersion** (*string*) – dependeeVersion
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

DELETE /services/{shortName}/{version}/dependees/{dependeeShortName}/{dependeeVersion}
deleteDependee

Parameters

- **dependeeShortName** (*string*) – dependeeShortName
- **dependeeVersion** (*string*) – dependeeVersion
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

GET /services/{shortName}/{version}/dependencyGraph
getDependencyGraph

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[]**.href (*string*) –
- **_links[]**.templated (*boolean*) –

- **micoServiceDependencyGraphEdgeList [] .sourceShortName** (*string*) – The short name of the source MicoService.
- **micoServiceDependencyGraphEdgeList [] .sourceVersion** (*string*) – The version of the source MicoService.
- **micoServiceDependencyGraphEdgeList [] .targetShortName** (*string*) – The short name of the target MicoService.
- **micoServiceDependencyGraphEdgeList [] .targetVersion** (*string*) – The version of the target MicoService.
- **micoServices [] .contact** (*string*) – Human readable contact information for support purposes.
- **micoServices [] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **micoServices [] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **micoServices [] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **micoServices [] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **micoServices [] .name** (*string*) – A human readable name of the MicoService. (required)
- **micoServices [] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **micoServices [] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **micoServices [] .shortName** (*string*) – A unique name of the MicoService. (required)
- **micoServices [] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

GET /services/{shortName}/{version}/dependers
getDependers

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_embedded [] ._links [] .href** (*string*) –
- **_embedded [] ._links [] .templated** (*boolean*) –

- **__embedded[] .contact** (*string*) – Human readable contact information for support purposes.
- **__embedded[] .description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **__embedded[] .dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **__embedded[] .dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **__embedded[] .gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **__embedded[] .name** (*string*) – A human readable name of the MicoService. (required)
- **__embedded[] .owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **__embedded[] .serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **__embedded[] .shortName** (*string*) – A unique name of the MicoService. (required)
- **__embedded[] .version** (*string*) – The version of this service. Refers to GitHub release tag. (required)
- **__links[] .href** (*string*) –
- **__links[] .templated** (*boolean*) –

GET /services/{shortName}/{version}/interfaces
getInterfacesOfService

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **__embedded[] .__links[] .href** (*string*) –
- **__embedded[] .__links[] .templated** (*boolean*) –
- **__embedded[] .description** (*string*) – Human readable description of this service interface. Null values are skipped.
- **__embedded[] .ports[] .port** (*integer*) – The port number of the externally exposed port. (required)
- **__embedded[] .ports[] .targetPort** (*integer*) – The port inside the container. (required)

- **_embedded[] .ports[] .type** (*string*) – The type (protocol) of the port. TCP or UDP. (required)
- **_embedded[] .protocol** (*string*) – The protocol of this interface.
- **_embedded[] .publicDns** (*string*) – The public DNS.
- **_embedded[] .serviceName** (*string*) – The name of this MicoServiceInterface (required)
- **_embedded[] .transportProtocol** (*string*) – The transport protocol of this interface.
- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –

POST /services/{shortName}/{version}/interfaces
createServiceInterface

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **description** (*string*) – Human readable description of this service interface. Null values are skipped.
- **ports[] .port** (*integer*) – The port number of the externally exposed port. (required)
- **ports[] .targetPort** (*integer*) – The port inside the container. (required)
- **ports[] .type** (*string*) – The type (protocol) of the port. TCP or UDP. (required)
- **protocol** (*string*) – The protocol of this interface.
- **publicDns** (*string*) – The public DNS.
- **serviceName** (*string*) – The name of this MicoServiceInterface (required)
- **transportProtocol** (*string*) – The transport protocol of this interface.

Status Codes

- **200 OK** – OK
- **201 Created** – Created
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **description** (*string*) – Human readable description of this service interface. Null values are skipped.

- **ports[] .port** (*integer*) – The port number of the externally exposed port. (required)
- **ports[] .targetPort** (*integer*) – The port inside the container. (required)
- **ports[] .type** (*string*) – The type (protocol) of the port. TCP or UDP. (required)
- **protocol** (*string*) – The protocol of this interface.
- **publicDns** (*string*) – The public DNS.
- **serviceInterfaceName** (*string*) – The name of this MicoServiceInterface (required)
- **transportProtocol** (*string*) – The transport protocol of this interface.

GET /services/{shortName}/{version}/interfaces/{serviceInterfaceName}
getInterfaceByName

Parameters

- **serviceInterfaceName** (*string*) – serviceInterfaceName
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **description** (*string*) – Human readable description of this service interface. Null values are skipped.
- **ports[] .port** (*integer*) – The port number of the externally exposed port. (required)
- **ports[] .targetPort** (*integer*) – The port inside the container. (required)
- **ports[] .type** (*string*) – The type (protocol) of the port. TCP or UDP. (required)
- **protocol** (*string*) – The protocol of this interface.
- **publicDns** (*string*) – The public DNS.
- **serviceInterfaceName** (*string*) – The name of this MicoServiceInterface (required)
- **transportProtocol** (*string*) – The transport protocol of this interface.

PUT /services/{shortName}/{version}/interfaces/{serviceInterfaceName}
updateServiceInterface

Parameters

- **serviceInterfaceName** (*string*) – serviceInterfaceName

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **description** (*string*) – Human readable description of this service interface. Null values are skipped.
- **ports[] .port** (*integer*) – The port number of the externally exposed port. (required)
- **ports[] .targetPort** (*integer*) – The port inside the container. (required)
- **ports[] .type** (*string*) – The type (protocol) of the port. TCP or UDP. (required)
- **protocol** (*string*) – The protocol of this interface.
- **publicDns** (*string*) – The public DNS.
- **serviceName** (*string*) – The name of this MicoServiceInterface (required)
- **transportProtocol** (*string*) – The transport protocol of this interface.

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **description** (*string*) – Human readable description of this service interface. Null values are skipped.
- **ports[] .port** (*integer*) – The port number of the externally exposed port. (required)
- **ports[] .targetPort** (*integer*) – The port inside the container. (required)
- **ports[] .type** (*string*) – The type (protocol) of the port. TCP or UDP. (required)
- **protocol** (*string*) – The protocol of this interface.
- **publicDns** (*string*) – The public DNS.
- **serviceName** (*string*) – The name of this MicoServiceInterface (required)
- **transportProtocol** (*string*) – The transport protocol of this interface.

DELETE /services/{shortName}/{version}/interfaces/{serviceName}
deleteServiceInterface

Parameters

- **serviceName** (*string*) – serviceName

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 204 No Content – No Content
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden

GET /services/{shortName}/{version}/interfaces/{serviceInterfaceName}/publicIP
getInterfacePublicIpByName

Parameters

- **serviceInterfaceName** (*string*) – serviceInterfaceName
- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- 200 OK – OK
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- [] (*string*) –

POST /services/{shortName}/{version}/promote
promoteService

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Request JSON Object

- **version** (*string*) – The version (in semantic version format). (required)

Status Codes

- 200 OK – OK
- 201 Created – Created
- 401 Unauthorized – Unauthorized
- 403 Forbidden – Forbidden
- 404 Not Found – Not Found

Response JSON Object

- **_links**[].**href** (*string*) –
- **_links**[].**templated** (*boolean*) –
- **contact** (*string*) – Human readable contact information for support purposes.

- **description** (*string*) – Human readable description of this service. Is allowed to be empty (default). Null values are skipped. (required)
- **dockerImageUri** (*string*) – The fully qualified URI to the image on DockerHub.
- **dockerfilePath** (*string*) – The relative path to the Dockerfile.
- **gitCloneUrl** (*string*) – The URL used for a git clone of a GitHub repository, to clone the current master branch.
- **name** (*string*) – A human readable name of the MicoService. (required)
- **owner** (*string*) – Human readable information for the services owner, who is responsible for this service.
- **serviceCrawlingOrigin** (*string*) – Indicates where this service originates from. (read only)
- **shortName** (*string*) – A unique name of the MicoService. (required)
- **version** (*string*) – The version of this service. Refers to GitHub release tag. (required)

GET /services/{shortName}/{version}/status
getStatusOfService

Parameters

- **shortName** (*string*) – shortName
- **version** (*string*) – version

Status Codes

- **200 OK** – OK
- **401 Unauthorized** – Unauthorized
- **403 Forbidden** – Forbidden
- **404 Not Found** – Not Found

Response JSON Object

- **_links[] .href** (*string*) –
- **_links[] .templated** (*boolean*) –
- **applicationsUsingThisService[] .contact** (*string*) – Human readable contact information for support purposes.
- **applicationsUsingThisService[] .deploymentStatus** (*string*) – Holds the current deployment status of this application. (read only)
- **applicationsUsingThisService[] .description** (*string*) – Human readable description of this application. Is allowed to be empty (default). Null values are skipped. (required)
- **applicationsUsingThisService[] .name** (*string*) – Human readable name of the application. (required)
- **applicationsUsingThisService[] .owner** (*string*) – Human readable information for the application owner, who is responsible for this application.
- **applicationsUsingThisService[] .shortName** (*string*) – Unique short name of the application. (required)

- **applicationsUsingThisService[] .version** (*string*) – The version of this application. (required)
- **availableReplicas** (*integer*) – Counter for the number of replicas of the corresponding MicoService that are actually available.
- **errorMessages[]** (*string*) –
- **interfacesInformation[] .externalIps[]** (*string*) –
- **interfacesInformation[] .name** (*string*) – Name of the MicoServiceInterface.
- **name** (*string*) – Name of the MicoService.
- **nodeMetrics[] .averageCpuLoad** (*integer*) – The average CPU load of all pods of one MicoService running on this Node.
- **nodeMetrics[] .averageMemoryUsage** (*integer*) – The average memory usage of all pods of one MicoService running on this Node.
- **nodeMetrics[] .nodeName** (*string*) – Name of the Kubernetes Node.
- **podsInformation[] .hostIp** (*string*) – IP address of the host to which the pod is assigned.
- **podsInformation[] .metrics .available** (*boolean*) – Indicates if a pod is available.
- **podsInformation[] .metrics .cpuLoad** (*integer*) – CPU load of a pod.
- **podsInformation[] .metrics .memoryUsage** (*integer*) – Memory usage of a pod.
- **podsInformation[] .nodeName** (*string*) – Name of the node the pod is running on.
- **podsInformation[] .phase** (*string*) – States the lifecycle of a pod: Pending, Running, Succeeded, Failed, Unknown, Completed, CrashLoopBackOff
- **podsInformation[] .podName** (*string*) – Name of the Pod.
- **podsInformation[] .restarts** (*integer*) – Counter for restarts of all containers of a pod.
- **podsInformation[] .startTime** (*string*) – The point in time the pod has been started.
- **requestedReplicas** (*integer*) – Counter for the number of replicas of the corresponding MicoService that should be available.
- **shortName** (*string*) – Short name of the MicoService.
- **version** (*string*) – Version of the MicoService.

5.3 Method Details (specification)

Specification for how the api should look like.

GET /services/

Example Response:

```
[
  {
    "id": "...",
    "shortName": "hello-world",
    "version": "0.0.1",
    "name": "Hello World",
    "description": "...",
  },
  {
    "id": "...",
    "shortName": "foo-bar",
    "version": "0.0.14a",
    "name": "FooBar",
    "description": "...",
  }
]
```

POST /services/
Example Request:

```
{
  "name": "Hello World",
  "shortName": "hello-world",
  "version": "0.0.1",
  "description": "...",
}
```

Example Response:

```
{
  "id": "...",
  "name": "Hello World",
  "shortName": "hello-world",
  "version": "0.0.1",
  "description": "...",
  "...": "...",
}
```

GET /services/ (str: shortName) /

GET /services/ (str: shortName) /
str: version/ Example Response:

```
{
  "id": "...",
  "name": "Hello World",
  "shortName": "hello-world",
  "version": "0.0.1",
  "description": "...",
  "...": "...",
  "interfaces": [],
  "dependees": [ "// services this service depends on" ],
  "dependers": [ "// services that depend on this service" ],
}
```

PUT /services/ (str: shortName) /
str: version/

```

GET /services/(str: shortName) /
  str: version/interfaces/
POST /services/(str: shortName) /
  str: version/interfaces/
GET /services/(str: shortName) /
  str: version/interfaces/str: shortName
PUT /services/(str: shortName) /
  str: version/interfaces/str: shortName
DELETE /services/(str: shortName) /
  str: version/interfaces/str: shortName
GET /services/(str: shortName) /
  str: version/dependees/
POST /services/(str: shortName) /
  str: version/dependees/
DELETE /services/(str: shortName) /
  str: version/dependees/str: shortName
GET /services/(str: shortName) /
  str: version/dependers/
POST /services/(str: shortName) /
  str: version/dependers/
DELETE /services/(str: shortName) /
  str: version/dependers/str: shortName
GET /services/(str: shortName) /
  str: version/status/
GET /services/import/
POST /services/import/(str: type) /

```

Example Response

```

HTTP/1.1 202 Accepted
Location: /jobs/12345

```

```
GET /jobs/(str: id) /
```

Example Response

```

GET /jobs/12345
HTTP/1.1 200 Ok
{
  "type": "IMPORT",
  "status": "PENDING",
  ...
  "_links": {
    "self": { "href": "/jobs/12345" },
    "cancel": { "method": "DELETE", "href": "/jobs/12345" },
  },
}

```

Example Response (Finished Job)

```
HTTP/1.1 303 See Other
Location: /services/{shortName}/{version} // depends on "type"
```

Note: /applications/ has all service endpoints as well...

GET /applications/ (str: *shortName*) /
str: *version*/

```
{
  "name": "...",
  "description": "...",
  "...": "...",
  "endpoint": {
    // just one "interface description"
  },
  // no dependers, dependees
  "services": [{
    // Service Object
  }]
}
```

POST /applications/ (str: *shortName*) /
str: *version*/deploy/

GET /cluster/*

6.1 MICO Setup

This section covers the setup of MICO. The current Kubernetes development environment is deployed to a Kubernetes cluster created with the Azure Kubernetes Service (AKS). It is described in [Azure Kubernetes Service](#).

6.1.1 Installation

Script:

To install MICO to your Kubernetes cluster you can use the interactive setup script `install/kubernetes/setup.sh` from the [MICO repository](#). It will install all MICO components and its requirements to your cluster that is configured through `kubectl`. You can find more information about the script in the official [README](#).

Structure:

The development of MICO is driven by the [monolith-first strategy](#).

MICO consists of

- Frontend: `mico-admin`
- Backend: `mico-core`

It requires the third-party components:

- Neo4j
- Knative Build
- Prometheus
- kube-state-metrics

For each component there is an own Kubernetes configuration file (YAML) placed in the directory `install/kubernetes`. They create the following namespace and deployment structure:

- `kube-system`

- kubernetes-dashboard
 - kube-state-metrics
 - metrics-server
 - ...
- mico-system
 - mico-admin (*frontend*)
 - mico-core (*backend*)
 - neo4j-core (*database*)
 - redis (*database*)
- mico-build-bot
- mico-workspace
- knative-build
 - build-controller
 - build-webhook
- monitoring
 - prometheus-core
 - kube-state-metrics
 - alertmanager
- kafka
 - kafka
 - pzoo
 - zoo
- openfaas
 - gateway
 - faas-idler
- openfaas-fn
 - custom functions

The most of the mentioned components are Kubernetes *Deployment* resources. `neo4j-core`, `redis`, `kafka`, `pzoo` and `zoo` are Kubernetes *StatefulSet* resources to be able to store the data persistently. The Neo4j is a graph database and is used to store all MICO resources. Redis is a key-value database and is used to store the background jobs.

At the beginning the namespaces `mico-build-bot` and `mico-workspace` are empty. `mico-build-bot` is used for the build processing of the Docker images using *Knative Build* and *Kaniko*. `mico-workspace` is used for the deployment of MICO services. Via labels they are composed to MICO applications.

Authentication:

`mico-core` requires super-user access to perform privileged actions on other resources. To give full control over every resource in the cluster and in all namespaces we use a `ClusterRoleBinding`. The configuration file `mico-cluster-admin.yaml` creates such a `ClusterRoleBinding` for the ServiceAccount “default” in the namespace `mico-system` that is used by `mico-core`.

6.1.2 Deployment

A deployment of a MICO application contains several steps:

1. Build container images for all included MICO services
2. Create Kubernetes *Deployments* and Kubernetes *Services*:
 - MICO Service → Kubernetes Deployment
 - MICO Service Interface → Kubernetes Service
3. Create interface connections

1. Build

Container images are build by `kaniko` as part of the MICO system inside the Kubernetes cluster. Kaniko itself runs as a container and is triggered by `Knative Build`. Kaniko builds the image based on the given Git clone URL, the Git revision (in our case always the release tag) and the path to the Dockerfile inside the Git repository (per default “Dockerfile”). After the build Kaniko pushes the image to the given Docker registry (per default DockerHub). The Build resources and the build Pods are created in the namespace `mico-build-bot`. The build name is created by `mico-core` based on the name of the MICO service and its version with the prefix `build-` (e.g. `build-hello-v1-0-0`). The build pods get the same name with a random suffix (e.g. `build-hello-v1-0-0-pod-50b30d`). The name is important because subsequent builds reuse existing Build resources if they have the same name.

`mico-core` waits 10 seconds and checks if the build is finished. Subsequent checks follow in a 5 seconds cycle. We consider a build as finished if the build pod has the phase `Succeeded` or `Failed`. If the build was successful the specified image name is stored as part of the MICO service entity in the database.

2. Kubernetes Resources

After all builds are finished, the required Kubernetes resources will be created or updated:

- MICO service → Kubernetes *Deployment*
- MICO service interface → Kubernetes *Service*

Names:

The names that are used for the Kubernetes resources are based on the names of the respective MICO resources (MICO service or MICO service interface) plus an additional random string with 8 alphanumeric characters. Examples:

- Kubernetes *Deployment* (name of the MICO service is `hello`): `hello-q1uhya8o`
- Kubernetes *Service* (name of the MICO service interface is `http`): `http-aopii3eg`

Labels:

Each Kubernetes *Deployment* and Kubernetes *Service* gets the following 4 MICO specific labels:

- `ust.mico/name`:
 - The name of a MICO service
 - Example value: “hello”
- `ust.mico/version`
 - The current version of a MICO service (semantic version)
 - Example value: “v1.0.0”

- `ust.mico/interface`
 - The name of a MICO service interface
 - Example value: “http”
- `ust.mico/instance`
 - The unique name of the MICO resource to be able to identify the associated Kubernetes resources
 - Example value: “hello-s6djsekw”

Scaling:

If a MICO service is already deployed (by another application) a scale out is performed (increasing the replicas) instead of recreating the Kubernetes *Deployment*.

Creation:

For the creation of a the Kubernetes *Deployment* the following service deployment information are used:

- Replicas
- Labels
- Environment Variables
- Image Pull Policy

Database:

The name of the created Kubernetes *Deployment* and the Kubernetes *Services* are stored as part of the MICO service deployment information in the database. That’s necessary to know, whether the deployment was already performed with this specific service deployment information (that means that this *MicoApplication* triggered the deployment). If these information are not given, but the MICO service is already deployed, it’s considered to be deployed by another MICO application.

3. Interface Connection

To be able to connect different MICO services with each other, `mico-core` creates environment variables with the DNS name of the interface of the other MICO service to connect with. These environment variables are applied on the Kubernetes *Deployment* of the MICO service. The information that are required for such an interface connection is part of the service deployment information and consists of three properties:

- the short name of the MICO service to connect with (e.g. `spring-boot-realworld-example-app`)
- the name of the MICO service interface to connect with (e.g. `rest`)
- the name of the environment variable used by the frontend (e.g. `BACKEND_REST_API`)

6.2 Azure Kubernetes Service

The MICO development environment runs on the Azure Kubernetes Service (AKS).

6.2.1 First steps

To connect to the cluster you need two command-line tools:

- *kubectl*: [Install and Set Up kubectl](#)
- Azure CLI *az*: [Install the Azure CLI](#)

To be able to execute the cluster related commands used in this documentation set following environment variables:

```
export LOCATION=westeurope
export RESOURCE_GROUP=ust-mico
export CLUSTER_NAME=ust-mico-cluster-aks
export RESOURCE_GROUP_NODE=MC_$RESOURCE_GROUP\_$_CLUSTER_NAME\_$_LOCATION
```

Sign in to Azure ([more information](#)):

```
az login
```

Configure kubectl to connect to the cluster ([more information](#)):

```
az aks get-credentials --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --admin
```

If you use kubectl to connect to multiple clusters, it's useful to use multiple contexts. To switch to the automatically created context of the MICO cluster, use

```
kubectl config use-context $CLUSTER_NAME-admin
```

6.2.2 Cluster details

Current cluster:

- VM: Standard_B2ms (2 vCPUs, 8 GB RAM)
- Nodes: 3
- OS Disk Size: 30 GB
- Location: westeurope
- Kubernetes version: 1.13.5

Get the details for a managed Kubernetes cluster:

```
az aks show --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME
```

Costs:

With two nodes we currently consume about 4 € per day.

You can check the current costs either in the [Azure Portal](#) under “Visual Studio Enterprise - Cost analysis” or in the [subscription dashboard](#) of your Azure account.

6.2.3 Upgrade Kubernetes

Upgrade an Azure Kubernetes Service (AKS) cluster

Check if there are updates available:

```
az aks get-upgrades --name $CLUSTER_NAME --resource-group $RESOURCE_GROUP --output_
↪table
```

Upgrade Kubernetes to specific version:

```
az aks upgrade --name $CLUSTER_NAME --resource-group $RESOURCE_GROUP --kubernetes-
↪version 1.12.6
```

Confirm that the upgrade was successful:

```
az aks show --name $CLUSTER_NAME --resource-group $RESOURCE_GROUP --output table
```

6.2.4 Cluster creation

Resource group

Our self created resource group is named `ust-mico-resourcegroup`. Whenever a resource group is created, “the AKS resource provider automatically creates the second one during deployment, such as `MC_myResourceGroup_myAKSCluster_eastus`” ([FAQ AKS](#)). This second resource group contains all of the infrastructure resources associated with the cluster (e.g. Kubernetes node VMs, virtual networking, and storage). The automatically created second resource group is named `MC_ust-mico-resourcegroup_ust-mico-cluster_westeurope`. It is a concatenation of `MC_$RESOURCE_GROUP__$CLUSTER_NAME__$LOCATION`.

Create Azure Kubernetes Service (AKS)

Create AKS:

```
az aks create --resource-group $RESOURCE_GROUP \
--name $CLUSTER_NAME \
--generate-ssh-keys \
--kubernetes-version 1.13.5 \
--node-vm-size Standard_B2ms \
--node-count 2
```

For more information see [Quickstart: Deploy an Azure Kubernetes Service \(AKS\) cluster](#).

6.2.5 Cluster deletion

Delete cluster:

```
az aks delete --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME \
```

Redeploy an Azure Kubernetes Service (AKS) cluster

If an AKS cluster was deleted by accident or was removed and should be setup again.

Create AKS:

Use command to create an AKS cluster like described above in section *Create Azure Kubernetes Service (AKS)*.

Setup credentials again:

- on Jenkins VM:
 - remove old config file in `~/ .kube/` (only if new cluster has the same name as the old cluster)
 - get new credentials:

```
az aks get-credentials --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME
```

- copy the new generate config file from `~/ .kube/` to `/var/lib/jenkins:`

```
sudo cp ~/.kube/config /var/lib/jenkins
```

- set ownership of config file to jenkins user + jenkins group

```
sudo chown jenkins:jenkins /var/lib/jenkins/config
```

6.2.6 Cluster management

Open Kubernetes dashboard

```
az aks browse --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME
```

Grant full admin privileges to Dashboard's Service Account [more information](#):

```
kubectl create clusterrolebinding kubernetes-dashboard --clusterrole=cluster-admin --
↪serviceaccount=kube-system:kubernetes-dashboard \
&& kubectl label clusterrolebinding kubernetes-dashboard k8s-app=kubernetes-dashboard
```

WARNING: This configuration of a ClusterRoleBinding may lead to security issues, there is no additional authentication and the dashboard is publicly accessible! [Access the Kubernetes web dashboard in Azure Kubernetes Service \(AKS\)](#)

Public Access

Create static IP address for the MICO dashboard:

For a stable, public access to the MICO dashboard, a static IP address is recommended. An IP address in Azure with the name mico-dashboard-ip can be generated by:

```
az network public-ip create \
  --resource-group $RESOURCE_GROUP_NODE \
  --name mico-dashboard-ip \
  --allocation-method static
```

Get the assigned IP address:

```
az network public-ip show --resource-group $RESOURCE_GROUP_NODE --name mico-dashboard-
↪ip --query ipAddress --output tsv
```

The dashboard of mico-admin is accessible via <http://40.118.1.61/dashboard>.

Side note: It was not possible to expose the deployment mico-admin via the command `kubectl expose deployment mico-admin --external-ip=...` because it created a complete new IP address instead of just using the specified external IP address.

Create static IP address for the OpenFaaS Portal:

To get a static access to the OpenFaaS Portal, you can also create a static IP address for it:

```
az network public-ip create \
  --resource-group $RESOURCE_GROUP_NODE \
  --name mico-openfaas-portal-ip \
  --allocation-method static
```

Get the assigned IP address:

```
az network public-ip show --resource-group $RESOURCE_GROUP_NODE --name mico-openfaas-portal-ip --query ipAddress --output tsv
```

The OpenFaaS portal is accessible via <http://40.115.25.83:8080/ui>.

6.3 Google Kubernetes Engine

MICO was developed with the [Azure Kubernetes Service](#) but is also tested with the Google Kubernetes Engine.

6.3.1 First steps

To connect to the cluster you need two command-line tools:

- *kubectl*: [Install and Set Up kubectl](#)
- Google Cloud SDK *gcloud*: [Google Cloud SDK documentation](#)

Initialization:

Use `gcloud init` to get started and to create a new project (e.g. `ust-mico`).

Check if the correct project is chosen:

```
gcloud config list
```

Set your default project:

```
gcloud config set project ust-mico
```

Set your default [compute zone](#):

```
gcloud config set compute/zone europe-west3
```

Ensure that billing is enabled for the project: [Learn how to enable billing](#).

Configure `kubectl` to connect to the cluster ([more information](#)):

```
gcloud container clusters get-credentials ust-mico-cluster
```

If you use `kubectl` to connect to multiple clusters, it's useful to use multiple contexts. To switch to the automatically created context of the MICO cluster, use

```
kubectl config use-context gke_ust-mico_europe-west3_ust-mico-cluster
```

Check the current context:

```
kubectl config current-context
```

6.3.2 Cluster details

Current cluster:

- VM: `n1-standard-2` (2 vCPUs, 7.5 GB RAM)
- Nodes: 3 (1 per zone)

- OS Disk Size: 45 GB
- Location: westeurope
- Kubernetes version: 1.12.6

6.3.3 Cluster creation

Create cluster:

```
gcloud container --project "ust-mico" clusters create "ust-mico-cluster" --region
↪ "europe-west3" --cluster-version "1.12.6-gke.7" --machine-type "n1-standard-2" --
↪ num-nodes "1" --enable-cloud-logging --enable-cloud-monitoring --addons_
↪ KubernetesDashboard --enable-autoupgrade --enable-autorepair --maintenance-window
↪ "02:00"
```

For more information see [Creating a cluster](#).

Prerequisites for using Role-Based Access Control:

To be able to install MICO, it's required to grant the user the ability to create roles in Kubernetes. This is a prerequisite to use role-based access control on GKE. For more information see the [GKE instructions](#).

Grant permission for the current user:

```
kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --
↪ user $(gcloud config list account --format "value(core.account) ")
```

6.3.4 Cluster deletion

Delete cluster:

```
gcloud container --project "ust-mico" clusters delete "ust-mico-cluster"
```

6.3.5 Cluster management

Public Access

Create static IP address:

For a stable, public access to the MICO dashboard, a static IP address is recommended. This IP address with the name `mico-dashboard-ip` was generated by:

```
gcloud compute addresses create mico-dashboard-ip --region europe-west3
```

Get the assigned IP address:

```
gcloud compute addresses describe mico-dashboard-ip --region europe-west3
```

Public static IP address: 35.246.187.4

Dashboard:

The dashboard of `mico-admin` is accessible via <http://35.246.187.4/dashboard>.

6.4 Development environment

6.4.1 Configuration files

EditorConfig

We use EditorConfig to maintain consistent coding styles between different editors and IDEs. EditorConfig files are human readable and most IDEs support it without additional plugins, e.g. IntelliJIDEA. Nevertheless Eclipse and VisualStudio Code need additional plugins.

If you are using Eclipse, you can install the editorconfig-eclipse plugin. For VisualStudio Code you can install the EditorConfig for VS Code plugin.

The EditorConfig file is pushed in main directory of the mico repository.

```
# Editor configuration, see http://editorconfig.org
root = true

[*]
charset = utf-8
indent_style = space
indent_size = 4
insert_final_newline = true
trim_trailing_whitespace = true

[*.md]
max_line_length = off
trim_trailing_whitespace = false
```

Eclipse config

If you don't use EditorConfig, you can also use the following config for eclipse. You need to include it into eclipse config settings.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<profiles version="14">
  <profile kind="CodeFormatterProfile" name="MICO" version="14">
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_ellipsis"
    ↪value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_enum_
    ↪declarations" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_
    ↪annotation_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
    ↪allocation_expression" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_at_in_
    ↪annotation_type_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_for_
    ↪statment" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.new_lines_at_block_
    ↪boundaries" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
    ↪constructor_declaration_parameters" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.insert_new_line_for_
    ↪parameter" value="do not insert"/>
```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪on_package" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_method_
↪invocation" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_
↪in_enum_constant" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_after_imports" value=
↪"1"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_while" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.insert_new_line_before_
↪root_tags" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_
↪in_annotation_type_member_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪method_declaration_throws" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_switch_
↪statement" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.format_javadoc_comments"
↪value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.indentation.size" value="4"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_postfix_
↪operator" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_enum_
↪constant_declaration" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_for_
↪increments" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_type_
↪arguments" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_for_
↪inits" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_
↪anonymous_type_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_semicolon_in_
↪for" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.align_with_spaces" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.disabling_tag" value=
↪"@formatter:off"/>
    <setting id="org.eclipse.jdt.core.formatter.continuation_indentation" value="2
↪"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_enum_constants"
↪value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_before_imports" value=
↪"1"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_after_package" value=
↪"1"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_binary_operator
↪" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪multiple_local_declarations" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_if_while_
↪statement" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_enum_
↪constant" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_angle_
↪bracket_in_parameterized_type_reference" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.indent_root_tags" value=
↪"false"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.wrap_before_or_operator_multicatch
↪ " value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.enabling_tag" value="@formatter:on
↪ "/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_brace_
↪ in_block" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.count_line_length_from_
↪ starting_position" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_parenthesized_
↪ expression_in_return" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_throws_clause_in_
↪ method_declaration" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪ on_parameter" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.keep_then_statement_on_same_line"
↪ value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪ on_field" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪ explicit_constructor_call_arguments" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_block"
↪ value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_prefix_operator
↪ " value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_between_type_
↪ declarations" value="1"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_brace_
↪ in_array_initializer" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪ in_for" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪ in_catch" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_angle_
↪ bracket_in_type_arguments" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪ on_method" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪ in_switch" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_parameterized_type_
↪ references" value="0"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪ in_anonymous_type_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪ in_parenthesized_expression" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪ on_enum_constant" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.never_indent_line_comments_on_
↪ first_column" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_and_in_type_
↪ parameter" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_for_
↪ inits" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.indent_statements_compare_to_block
↪ " value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_anonymous_type_
↪ declaration" value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_question_in_
↪ wildcard" value="do not insert"/>

```

(continues on next page)

(continued from previous page)

```

<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪method_invocation_arguments" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_switch" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.align_tags_descriptions_
↪grouped" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.comment.line_length" value="80"/>
<setting id="org.eclipse.jdt.core.formatter.use_on_off_tags" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_
↪brackets_in_array_allocation_expression" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_enum_constant" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_paren_
↪in_method_invocation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_assignment_
↪operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_type_declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_for" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.preserve_white_space_
↪between_code_and_line_comments" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪on_local_variable" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_method_
↪declaration" value="end_of_line"/>
<setting id="org.eclipse.jdt.core.formatter.align_variable_declarations_on_
↪columns" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_method_invocation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_union_type_in_
↪multicatch" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_for" ↪
↪value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.number_of_blank_lines_at_
↪beginning_of_method_body" value="0"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_angle_
↪bracket_in_type_arguments" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.keep_else_statement_on_same_line" ↪
↪value="false"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_binary_expression" ↪
↪value="16"/>
<setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_catch_
↪clause" value="common_lines"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪parameterized_type_reference" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪array_initializer" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪multiple_field_declarations" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_
↪explicit_constructor_call" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_
↪to_annotation_declaration_header" value="true"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪superinterfaces" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_
↪default" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_question_in_
↪conditional" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_block" value=
↪"end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_constructor_
↪declaration" value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_lambda_body" ↪
↪value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.compact_else_if" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_type_
↪parameters" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_catch" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_method_invocation" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.put_empty_statement_on_new_line" ↪
↪value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_parameters_in_
↪constructor_declaration" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_type_parameters" ↪
↪value="0"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪method_invocation_arguments" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_method_
↪invocation" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_throws_clause_in_
↪constructor_declaration" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_compact_loops" ↪
↪value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.clear_blank_lines_in_
↪block_comment" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_catch_in_
↪try_statement" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_try" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.keep_simple_for_body_on_same_line
↪" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_at_end_of_file_if_
↪missing" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.clear_blank_lines_in_
↪javadoc_comment" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_array_
↪initializer" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_binary_
↪operator" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_unary_operator
↪" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_expressions_in_
↪array_initializer" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.format_line_comment_starting_on_
↪first_column" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.number_of_empty_lines_to_preserve
↪" value="1"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_
↪annotation" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_case"
↪value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_ellipsis"
↪value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_semicolon_in_
↪try_resources" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_assert
↪" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_if" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_type_
↪arguments" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_and_in_type_
↪parameter" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_type_
↪declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_parenthesized_expression" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.format_line_comments"
↪value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_
↪labeled_statement" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.align_type_members_on_columns"
↪value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_assignment" value="0
↪"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_module_statements"
↪value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_method_
↪body" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_
↪to_type_header" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_
↪in_method_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.align_tags_names_
↪descriptions" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_enum_constant" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_superinterfaces_in_
↪type_declaration" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_before_first_class_
↪body_declaration" value="0"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_conditional_
↪expression" value="80"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_closing_
↪brace_in_array_initializer" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪constructor_declaration_parameters" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.format_guardian_clause_on_one_line
↪" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_if" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.align_assignment_statements_on_
↪columns" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_annotation_
↪on_type" value="insert"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_block" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_enum_
↪declaration" value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_block_in_case"
↪value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_constructor_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.format_header" value=
↪"false"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_
↪allocation_expression" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_method_invocation" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_while" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_switch" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_method_declaration"
↪value="0"/>
    <setting id="org.eclipse.jdt.core.formatter.join_wrapped_lines" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_parens_
↪in_constructor_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.wrap_before_conditional_operator"
↪value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.indent_switchstatements_compare_
↪to_cases" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_
↪bracket_in_array_allocation_expression" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_synchronized" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.align_fields_grouping_blank_lines
↪" value="2147483647"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.new_lines_at_javadoc_
↪boundaries" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_annotation_
↪type_declaration" value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_for"
↪value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_resources_in_try"
↪value="80"/>
    <setting id="org.eclipse.jdt.core.formatter.use_tabs_only_for_leading_
↪indentations" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_try_
↪clause" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_selector_in_method_
↪invocation" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.never_indent_block_comments_on_
↪first_column" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_synchronized" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪constructor_declaration_throws" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.tabulation.size" value="4"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_enum_
↪constant" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪allocation_expression" value="insert"/>

```

(continues on next page)

(continued from previous page)

```

<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_
↪bracket_in_array_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_colon_in_
↪conditional" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_source_code" value=
↪"true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_array_initializer" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_try" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_semicolon_in_
↪try_resources" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_field" value="0
↪"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_at_in_
↪annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.continuation_indentation_for_
↪array_initializer" value="2"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_question_in_
↪wildcard" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_method" value=
↪"1"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_superclass_in_type_
↪declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_superinterfaces_in_
↪enum_declaration" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_parenthesized_
↪expression_in_throw" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.wrap_before_assignment_operator"
↪value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_
↪labeled_statement" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.brace_position_for_switch" value=
↪"end_of_line"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪superinterfaces" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪method_declaration_parameters" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_type_
↪annotation" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_brace_
↪in_array_initializer" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_parenthesized_expression" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.comment.format_html" value="true"/
↪>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_at_in_
↪annotation_type_declaration" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_angle_
↪bracket_in_type_parameters" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_method_
↪declaration" value="common_lines"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_compact_if" value=
↪"16"/>
<setting id="org.eclipse.jdt.core.formatter.indent_empty_lines" value="false"/
↪>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_type_arguments"
↪value="0"/>

```

(continues on next page)

(continued from previous page)

```

<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪parameterized_type_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_unary_operator
↪" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_enum_constant" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_
↪annotation" value="0"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_enum_
↪declarations" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.keep_empty_array_initializer_on_
↪one_line" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.indent_switchstatements_compare_
↪to_switch" value="false"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_else_in_if_
↪statement" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_assignment_
↪operator" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_constructor_declaration" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_new_chunk" ↪
↪value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_label" ↪
↪value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_
↪to_enum_declaration_header" value="true"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_
↪bracket_in_array_allocation_expression" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_constructor_declaration" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_
↪conditional" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_angle_
↪bracket_in_parameterized_type_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪method_declaration_parameters" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_angle_
↪bracket_in_type_arguments" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_cast" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_
↪assert" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.blank_lines_before_member_type" ↪
↪value="1"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_while_in_
↪do_statement" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_
↪bracket_in_array_type_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_angle_
↪bracket_in_parameterized_type_reference" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.alignment_for_arguments_in_
↪qualified_allocation_expression" value="16"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_after_opening_
↪brace_in_array_initializer" value="do not insert"/>
<setting id="org.eclipse.jdt.core.formatter.insert_new_line_in_empty_enum_
↪declaration" value="insert"/>
<setting id="org.eclipse.jdt.core.formatter.indent_breaks_compare_to_cases" ↪
↪value="true"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_method_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_if" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_semicolon"
↪value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_postfix_
↪operator" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_try" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_angle_
↪bracket_in_type_arguments" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_cast" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.format_block_comments"
↪value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_lambda_arrow"
↪value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_method_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.keep_imple_if_on_one_line" value=
↪"false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_enum_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_parameters_in_
↪method_declaration" value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_between_brackets_in_
↪array_type_reference" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_angle_
↪bracket_in_type_parameters" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_semicolon_in_
↪for" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪method_declaration_throws" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_
↪bracket_in_array_allocation_expression" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.indent_statements_compare_to_body
↪" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_multiple_fields"
↪value="16"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_enum_
↪constant_arguments" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.keep_simple_while_body_on_same_
↪line" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_prefix_
↪operator" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_array_
↪initializer" value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.wrap_before_binary_operator"
↪value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_method_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_type_
↪parameters" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_catch" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_
↪bracket_in_array_reference" value="do not insert"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_comma_in_
↪annotation" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_enum_
↪constant_arguments" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.parentheses_positions_in_lambda_
↪declaration" value="common_lines"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_between_empty_braces_
↪in_array_initializer" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_colon_in_case
↪" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪multiple_local_declarations" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.keep_simple_do_while_body_on_same_
↪line" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_annotation_type_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_
↪bracket_in_array_reference" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_method_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.wrap_outer_expressions_when_nested
↪" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_closing_paren_
↪in_cast" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_enum_constant" ↪
↪value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.brace_position_for_type_
↪declaration" value="end_of_line"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_before_package" value=
↪"0"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_for" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_synchronized" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_for_
↪increments" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪in_annotation_type_member_declaration" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.alignment_for_expressions_in_for_
↪loop_header" value="0"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪in_while" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_enum_constant" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪explicit_constructor_call_arguments" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_paren_
↪in_annotation" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_angle_
↪bracket_in_type_parameters" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.indent_body_declarations_compare_
↪to_enum_constant_header" value="true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_lambda_arrow" ↪
↪value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_brace_
↪in_constructor_declaration" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪constructor_declaration_throws" value="do not insert"/>

```

(continues on next page)

(continued from previous page)

```

    <setting id="org.eclipse.jdt.core.formatter.join_lines_in_comments" value=
↪ "true"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_closing_angle_
↪ bracket_in_type_parameters" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_question_in_
↪ conditional" value="insert"/>
    <setting id="org.eclipse.jdt.core.formatter.comment.indent_parameter_
↪ description" value="false"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_new_line_before_finally_in_
↪ try_statement" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.tabulation.char" value="space"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_comma_in_
↪ multiple_field_declarations" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.blank_lines_between_import_groups
↪ " value="1"/>
    <setting id="org.eclipse.jdt.core.formatter.lineSplit" value="120"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_after_opening_paren_
↪ in_annotation" value="do not insert"/>
    <setting id="org.eclipse.jdt.core.formatter.insert_space_before_opening_paren_
↪ in_switch" value="insert"/>
  </profile>
</profiles>

```

VisualStudio Code

If you don't use the EditorConfig plugin for VS Code, you can set the editor configurations on File, Preferences, Settings, Workspace Settings. Replace the following parts in the config.json

```

// The number of spaces a tab is equal to. This setting is overridden
// based on the file contents when `editor.detectIndentation` is true.
"editor.tabSize": 4,

// Insert spaces when pressing Tab. This setting is overridden
// based on the file contents when `editor.detectIndentation` is true.
"editor.insertSpaces": true,

// When opening a file, `editor.tabSize` and `editor.insertSpaces`
// will be detected based on the file contents. Set to false to keep
// the values you've explicitly set, above.
"editor.detectIndentation": false

```

6.4.2 Install Lombok Plugin

To reduce the volume of single java classes, Lombok is used to inject code via annotations. Go to [Project Lombok](#) and follow the install instructions for your desired IDE.

6.5 CI/CD Pipeline

Describes setup of a Continuous Integration and Continuous Deployment Pipeline with Kubernetes, Jenkins, and DockerHub.

6.5.1 Prerequisites

- Install Docker (on Ubuntu: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>)
- Install unzip `sudo apt-get install unzip`
- **Install Maven**

- `sudo mkdir /opt/maven/`
- `cd /opt/maven/`
- `wget http://ftp.wayne.edu/apache/maven/maven-3/<VERSION>/binaries/apache-maven-<VERSION>-bin.zip`
- `unzip apache-maven-<VERSION>`
- `sudo nano /etc/profile.d/maven.sh`
- **Add the following lines to maven.sh:**

```
export JAVA_HOME=<PATH_TO_YOUR_JAVA_HOME>
export M2_HOME=/opt/maven
export MAVEN_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}
```

- `sudo source /etc/profile.d/maven.sh`

6.5.2 Kubernetes Setup

Create a Kubernetes cluster with *Azure Kubernetes Service (AKS)* like described in [setup/Azure Kubernetes Service](#).

Note that there might be multiple Kubernetes configuration files (spethso, root, jenkins). Ensure that the Kubernetes configuration file for Jenkins is created correctly in `/var/lib/jenkins/config`. This path is used in the Jenkinsfile as a parameter for the `kubectl` commands: `--kubeconfig /var/lib/jenkins/config`.

6.5.3 Jenkins Setup

- Deploy a Jenkins VM through the Azure Marketplace
- Connect to the VM via ssh to update Jenkins and other applications: `ssh spethso@ust-mico-jenkins.westeurope.cloudapp.azure.com`
- Create Jenkins credentials to access DockerHub
- Create a Github Webhook: <https://docs.microsoft.com/en-us/azure/aks/jenkins-continuous-deployment#create-a-github-webhook>
- **To be able to run the integration tests, DockerHub credentials with Base64 encoding are required. Provide them as enviro**

```
export DOCKERHUB_USERNAME_BASE64=*****
export DOCKERHUB_PASSWORD_BASE64=*****
```

- Install BlueOcean Plugins for a better Pipeline UI
- Install Git Server Plugin, Git Plugin, Git Plugin for BlueOcean and GitHub Plugin for BlueOcean
- **Under GitHub in Configuration add a new GitHub Server**
 - API URL: <https://api.github.com>

- Provide your GitHub Access Token as credentials
 - Enable manage hooks
- **Create a new Jenkins Pipeline project:** <https://jenkins.io/doc/book/pipeline/>
 - Enable “GitHub hook trigger for GITScm polling”
 - **Provide Pipeline definition as a “Pipeline script from SCM”**
 - * Use “Git” and add the repository URL to <https://github.com/UST-MICO/mico>
 - * Set the script path to `Jenkinsfile`
 - * Set leightweight checkout to only checkout the Jenkinsfile and not the complete repository

Pipeline using the Jenkinsfile

The Jenkinsfile is in the root folder of the repository and contains the pipeline stages which are executed. The pipeline is triggered on a push on the master branch.

- **Checkout Stage**
 - This stage performs the checkout of the repository
- **Docker Build Stage**
 - `mico-core` and `mico-admin` Dockerimages are built in parallel
- **Unit test stage**
 - There is an own Dockerfile for unittests which is built in this stage
 - The Dockerimage runs the unittests using maven
- **Integration test stage**
 - There is an own Dockerfile for integration tests which is built in this stage
 - The Dockerimage runs the integrationtests using maven
- **Push image stage**
 - In this stage the Dockerimages for `mico-core` and `mico-admin` are pushed to Dockerhub, both one time with `kube` and the build number and one time with `latest` tag
- **Deploy on Kubernetes stage**
 - `mico-core` and `mico-admin` are deployed in parallel on Kubernetes using the `kubectl set image`
- **Docker clean up stage**
 - All Dockerimages older than 10 days are deleted in this stage
- **Post actions**
 - After the normale pipeline stages the build status is sent to the `ust-mico` slack workspace.

Adjust heap size of JRE

- Open the file `/etc/default/jenkins`
- Search for `JAVA_ARGS= '-Xmx256m'` (default)
- Remove the `#` to uncomment the line

- Adjust the size to the desired value
- **Example:**

```
JAVA_ARGS="-Xmx3g"
```

6.5.4 Travis CI

- To check pull requests before merge into master there is a Travis CI which builds everything with Maven.
 - Travis runs unit tests, but no integration tests
 - At the current state, Travis CI builds mico-core and mico-admin, but runs only unit tests for mico-core
- URL <https://travis-ci.org/UST-MICO/mico>

6.6 Kubernetes

6.6.1 Minikube

Minikube can be used to run a single node Kubernetes cluster on your local machine.

Installation

Read the official instructions how to [Install Minikube](#).

On Windows 10

Pre-reqs:

- Hyper-V enabled

Installation with chocolatey:

```
choco install minikube -y
choco install kubernetes-cli -y
```

Network Configuration:

Using PowerShell (run as Administrator):

```
Get-NetAdapter
New-VMSwitch -Name "ExternalSwitch" -NetAdapterName "Ethernet 2" -AllowManagement
↪ $True
```

Important commands

Run minikube:

```
minikube start --vm-driver hyperv --hyperv-virtual-switch "ExternalSwitch"
```

Stop minikube:

```
minikube stop
```

Get IP address:

```
minikube ip
```

Open dashboard:

```
minikube dashboard
```

Update minikube:

```
minikube update-check
```

Cleaning up cluster:

```
minikube delete
```

SSH (Minikube command):

```
minikube ssh
```

SSH (alternative command on WSL):

```
chmod 0600 "$ (wslpath -u "$ (minikube ssh-key) ") "
ssh -a -i "$ (wslpath -u "$ (minikube ssh-key) ") " -l docker "$ (minikube ip) "
```

Knative installation on Minikube

The following instructions are based on the official GitHub wiki [Knative Install on Minikube](#).

Minikube has to be started with privileged rights (root / administrator). Important note: At least **8 GiB of free memory** are required. Also 4 vCPUs are required, otherwise later on some pods will fail (status “OutOfcpu”).

Before starting Minikube, ensure that a virtual network switch called “ExternalSwitch” exists. If you are using Mac OS or Linux you have to change the vm-driver. For Linux specify `--vm-driver=kvm2`. Omitting the `--vm-driver` option will use the default driver.

Start Minikube with Hyper-V:

```
minikube start --memory=8192 --cpus=4 \
  --kubernetes-version=v1.13.5 \
  --vm-driver=hyperv \
  --hyperv-virtual-switch "ExternalSwitch" \
  --bootstrapper=kubeadm \
  --extra-config=apiserver.enable-admission-plugins="LimitRanger,NamespaceExists,
↪NamespaceLifecycle,ResourceQuota,ServiceAccount,DefaultStorageClass,
↪MutatingAdmissionWebhook"
```

Confirm that your kubectl context is pointing to the new cluster:

```
kubectl config current-context
```

Install Istio:

```
# Install Istio
curl -L https://github.com/knative/serving/releases/download/v0.2.2/istio.yaml \
| sed 's/LoadBalancer/NodePort/' \
| kubectl apply --filename -

# Label the default namespace with istio-injection=enabled.
kubectl label namespace default istio-injection=enabled
```

Monitor the Istio components:

```
kubectl get pods --namespace istio-system --watch
```

Install Knative Serving (lite version without monitoring components):

```
curl -L https://github.com/knative/serving/releases/download/v0.2.2/release-lite.yaml \
| sed 's/LoadBalancer/NodePort/' \
| kubectl apply --filename -
```

Monitor the Knative components:

```
kubectl get pods --namespace knative-serving --watch
```

Helpful commands:

Get IP address and NodePort for the knative-ingressgateway:

```
echo $(minikube ip):$(kubectl get svc knative-ingressgateway --namespace istio-system \
--output 'jsonpath={.spec.ports[?(@.port==80)].nodePort}')
```

Troubleshooting

- minikube stop does not work, command hangs and finally crashes. As a workaround use SSH instead [minikube issue 2914](#)

Poweroff via SSH:

```
minikube ssh "sudo poweroff"
```

- Kubernetes cluster shutdown after a few minutes MiniKube VM still running. Under Windows 10 the Hyper-V functionality “Dynamic Memory” could be the problem [minikube issue 2326](#)

Disable Dynamic RAM allocation in Hyper-V:

```
Set-VMemory minikube -DynamicMemoryEnabled $false
```

- If there is any problem with Hyper-V on Windows, you can try to restart the Hyper-V Virtual Machine Management service

Restarting Hyper-V Virtual Machine Management service:

```
net.exe stop vmms
net.exe start vmms
```

- If you have problems to start Minikube, you can delete the cache (everything is lost!)

On Linux:

```
rm -rf ~/.minikube
```

On Windows:

```
rm -rf %USERPROFILE%\.minikube
```

6.6.2 Azure

Azure CLI

Install: Install Azure CLI with apt

```
AZ_REPO=$(lsb_release -cs) && \
echo "deb [arch=amd64] https://packages.microsoft.com/repos/azure-cli/ $AZ_REPO main" \
->| sudo tee /etc/apt/sources.list.d/azure-cli.list

curl -L https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

sudo apt-get update && sudo apt-get install apt-transport-https azure-cli
```

Login:

```
az login
```

Azure Kubernetes Service

Knative test installation

Following instructions are based on the GitHub wiki [Knative Install on Azure Kubernetes Service \(AKS\)](#).

Create environment variables:

```
export LOCATION=westeurope
export RESOURCE_GROUP=ust-mico-knative-test-group
export CLUSTER_NAME=knative-test-cluster
```

Create a resource group for AKS:

```
az group create --name $RESOURCE_GROUP --location $LOCATION
```

Create a Kubernetes cluster using AKS:

Enable AKS:

```
az provider register -n Microsoft.ContainerService
```

Find out what Kubernetes versions are currently available:

```
az aks get-versions --location $LOCATION --output table
```

Create the AKS cluster:

```
az aks create --resource-group $RESOURCE_GROUP \
--name $CLUSTER_NAME \
--generate-ssh-keys \
--kubernetes-version 1.13.5 \
--node-vm-size Standard_DS3_v2 \
--node-count 1
```

Information about the used VM Standard_DS3_v2 DSv2-series:

- vCPU: 4
- Memory: 14 GiB
- Temp storage: 28 GiB
- Costs [estimation](#): 0,229 € per hour (per node) -> 5,51 € per day (per node)

Default node code is 3. Manual scaling is possible (see [Scaling](#)).

Configure kubectl to use the new cluster:

```
az aks get-credentials --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --admin
```

Verify your cluster is up and running

```
kubectl get nodes
```

Installing Istio:

```
# Install Istio
kubectl apply --filename https://github.com/knative/serving/releases/download/v0.2.2/
↪istio.yaml

# Label the default namespace with istio-injection=enabled.
kubectl label namespace default istio-injection=enabled
```

Monitor the Istio components:

```
kubectl get pods --namespace istio-system --watch
```

Install Knative components:

Install either Knative Serving + Build or only Knative Build.

Installing Knative Serving and Build components:

```
# Install Knative and its dependencies:
kubectl apply --filename https://github.com/knative/serving/releases/download/v0.2.2/
↪release.yaml

# Monitor the Knative components:
kubectl get pods --namespace knative-serving --watch
kubectl get pods --namespace knative-build --watch
```

Installing Knative Build only:

```
# Install Knative Build and its dependencies:
kubectl apply --filename https://github.com/knative/serving/releases/download/v0.2.2/
↪build.yaml
```

(continues on next page)

(continued from previous page)

```
# Monitor the Knative components:
kubectl get pods --namespace knative-build --watch
```

Deployment:

How to deploy applications with Knative is described in chapter [Knative](#).

Scaling**Manual scaling of AKS nodes:**

```
az aks scale --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --node-count 1
```

SSH into AKS cluster

Connect with SSH to Azure Kubernetes Service (AKS) cluster nodes for maintenance or troubleshooting

```
az aks show --resource-group myResourceGroup --name myAKSCluster --query_
↪nodeResourceGroup -o tsv

az vm list --resource-group MC_ust-mico-knative_knative-cluster_westeurope -o table

az vm user update \
  --resource-group MC_ust-mico-knative_knative-cluster_westeurope \
  --name aks-nodepool1-35709218-0 \
  --username azureuser \
  --ssh-key-value ~/.ssh/id_rsa.pub

az vm list-ip-addresses --resource-group MC_ust-mico-knative_knative-cluster_
↪westeurope -o table

kubectl run -it --rm aks-ssh --image=debian

apt-get update && apt-get install openssh-client -y

kubectl cp ~/.ssh/id_rsa aks-ssh-66cf68f4c7-4pz6m:/id_rsa

chmod 0600 id_rsa

ssh -i id_rsa azureuser@10.240.0.4
```

Helpful commands**Cleaning up Kubernetes cluster:**

```
az aks delete --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --yes --no-wait
```

Delete cluster and context from kubectl config:

```
kubectl config delete-cluster $CLUSTER_NAME
kubectl config delete-context $CLUSTER_NAME-admin
```

Open Kubernetes Dashboard:

```
az aks browse --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME
```

Azure Container Registry (ACR)

All images are stored in DockerHub in public. ACR is not used anymore.

At the beginning of the development of MICO the Azure Container Registry (ACR) was used to store the container images of the MICO system components. Because DockerHub was used as the container registry for the images build by MICO anyway, there was no reason to use two container registry at the same time.

Preparation:

Set environment variable:

```
export ACR_NAME=ustmicoregistry
```

Login:

```
az acr login --name $ACR_NAME
```

Create Azure Container Registry (ACR)

Create ACR:

```
az acr create --resource-group $RESOURCE_GROUP \
--name $ACR_NAME \
--sku Basic
```

For more information see [Tutorial: Deploy and use Azure Container Registry](#).

Grant Kubernetes Cluster access to ACR

When you create an AKS cluster, Azure also creates a service principal. We will use this auto-generated service principal for authentication with the ACR registry.

Grant AKS read access to ACR:

```
#!/bin/bash

AKS_RESOURCE_GROUP=$RESOURCE_GROUP
AKS_CLUSTER_NAME=$CLUSTER_NAME
ACR_RESOURCE_GROUP=$RESOURCE_GROUP
ACR_NAME=$ACR_NAME

# Get the id of the service principal configured for AKS
CLIENT_ID=$(az aks show --resource-group $AKS_RESOURCE_GROUP --name $AKS_CLUSTER_NAME_
↪--query "servicePrincipalProfile.clientId" --output tsv)

# Get the ACR registry resource id
ACR_ID=$(az acr show --name $ACR_NAME --resource-group $ACR_RESOURCE_GROUP --query "id
↪" --output tsv)

# Create role assignment
az role assignment create --assignee $CLIENT_ID --role Reader --scope $ACR_ID
```

Manage ACR

List container images:

```
az acr repository list --name $ACR_NAME --output table
```

List image tags (latest 10 tags of repository `mico-core`):

```
az acr repository show-tags --name $ACR_NAME --repository mica-core --top 10 --
↳ orderby time_desc --output table
```

Get credentials:

```
az acr credential show --name $ACR_NAME
```

Get the fully qualified name of the ACR login server:

```
az acr list --resource-group $RESOURCE_GROUP --query "[].{acrLoginServer:loginServer}
↳ " --output table
```

Authenticate with Azure Container Registry from Azure Kubernetes Service

Get the id of the service principal configured for AKS:

```
az aks show --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --query
↳ "servicePrincipalProfile.clientId" --output tsv
```

Get the ACR registry resource id:

```
az acr show --name $ACR_NAME --resource-group $RESOURCE_GROUP --query "id" --output
↳ tsv
```

Static IP addresses

List all static IP addresses:

```
az network public-ip list
```

Show IP address of MICO-PublicIP:

```
az network public-ip show --resource-group $RESOURCE_GROUP_NODE --name MICO-PublicIP -
↳ --query ipAddress --output tsv
```

Create new static IP address with name MICO-PublicIP:

```
az network public-ip create --resource-group $RESOURCE_GROUP_NODE --name MICO-
↳ PublicIP --allocation-method static
```

Delete static IP address with name `kubernetes-aa7005...`:

```
az network public-ip delete -g $RESOURCE_GROUP -n kubernetes-
↳ aa70055e8fc8911e88c2ad233499c1fd
```

6.6.3 Istio

Version

As of time of writing, Istio 1.0.5 is the latest stable release. Istio 1.1 is available as a preliminary snapshot release. A first release candidate is expected later in January and a final build sometime in February ([Status of Istio 1.1](#)).

Installation

Prerequisites:

1. [Download the Istio release](#).
2. Perform the necessary [platform-specific setup](#).
3. Check the [Requirements for Pods and Services](#) on Pods and Services.
4. [Install the Helm client](#).
5. Istio by default uses `LoadBalancer` service object types. Some platforms do not support `LoadBalancer` service objects. For platforms lacking `LoadBalancer` support, install Istio with `NodePort` support instead with the flags `--set gateways.istio-ingressgateway.type=NodePort --set gateways.istio-egressgateway.type=NodePort` appended to the end of the Helm operation.

Custom installation - Telemetry

To begin with we want to have a minimal Istio installation that only has its telemetry features enabled.

Required components:

- Envoy + automatic sidecar injection
 - Reporting of telemetry (logs + metrics)
- Mixer
 - Collect telemetry data
 - Provides interfaces ([adapters](#)) for specific infrastructure backends that support predefined [templates](#). Examples: Prometheus (`Metric`), Fluentd (`Log Entry`), Stackdriver (`Log Entry`, `Metric`, `Trace Span`), ...
- Prometheus
 - Collect metrics
- Fluentd
 - Collect logs

We use [Helm](#) for templating only.

Helm parameters:

- `security.enabled=false`: Citadel won't be installed
- `ingress.enabled=false`: Ingress won't be installed
- `gateways.istio-ingressgateway.enabled=false`: Ingress gateway won't be installed
- `gateways.istio-egressgateway.enabled=false`: Egress gateway won't be installed
- `galley.enabled=false`: Galley won't be installed

- `sidecarInjectorWebhook.enabled=false`: Automatic sidecar-injector won't be installed
- `mixer.policy.enabled=false`: Mixer Policy won't be installed
- `mixer.telemetry.enabled=true`: Mixer Telemetry will be installed (default)
- `global.proxy.envoyStatsd.enabled=false`: Disable Statsd (default)
- `pilot.sidecar=false`?
- `prometheus.enabled=true`: Prometheus addon will be installed (default)
- `grafana.enabled=true`: Grafana addon will be installed
- `tracing.enabled=false`: Tracing(jaeger) addon will be installed
- `kiali.enabled=false`: Kiali addon will be installed

Render Istio's core components to a Kubernetes manifest:

```
helm template install/kubernetes/helm/istio --name istio --namespace istio-system \
--set security.enabled=false \
--set ingress.enabled=false \
--set gateways.istio-ingressgateway.enabled=false \
--set gateways.istio-egressgateway.enabled=false \
--set galley.enabled=false \
--set sidecarInjectorWebhook.enabled=false \
--set mixer.policy.enabled=false \
--set mixer.telemetry.enabled=true \
--set global.proxy.envoyStatsd.enabled=false \
--set pilot.sidecar=false \
--set prometheus.enabled=true \
--set grafana.enabled=true \
--set tracing.enabled=false \
--set kiali.enabled=false \
> $HOME/istio-telemetry.yaml
```

Install the components via the manifest::

```
kubectl create namespace istio-system
kubectl apply -f $HOME/istio-telemetry.yaml
```

References:

Sidecar injection

Label the default namespace with `istio-injection=enabled`:

```
kubectl label namespace default istio-injection=enabled
```

Check for which namespaces the sidecar injection is enabled:

```
kubectl get namespace -L istio-injection
```

Upgrade Istio

Upgrading Istio

1. Download the new Istio release and change directory to the new release directory.

2. Upgrade Istio's Custom Resource Definitions via `kubectl apply`, and wait a few seconds for the CRDs to be committed in the kube-apiserver:

```
kubectl apply -f install/kubernetes/helm/istio/templates/crds.yaml
```

3. Upgrade with Kubernetes rolling update:

```
helm template install/kubernetes/helm/istio --name istio \
  --namespace istio-system > install/kubernetes/istio.yaml
kubectl apply -f install/kubernetes/istio.yaml
```

Architecture

Simplifying Microservice Architecture With Envoy and Istio

Two components:

- **Control Plane** - Controls the Data plane and is responsible for configuring the proxies for traffic management, policy enforcement, and telemetry collection.
- **Data Plane** - Comprised of Envoy proxies deployed as sidecars in each of the pods. All the application traffic flows through the Envoy proxies.

Pilot - Provides service discovery for the Envoy sidecars, traffic management capabilities for intelligent routing and resiliency (timeouts, retries, circuit breakers)

Mixer - Platform independent component which enforces access control and usage policies across the service mesh, and collects telemetry data from the Envoy proxy and other services.

Citadel - Provides strong service-to-service and end-user authentication with built-in identity and credential management.

All traffic entering and leaving the Istio service mesh is routed via the **Ingress/Egress Controller**. By deploying an Envoy proxy in front of services, you can conduct A/B testing, deploy canary services, etc. for user-facing services. Envoy is injected into the service pods inside the data plane using `Istioctl kube-inject`.

The Envoy sidecar logically calls Mixer before each request to perform precondition checks, and after each request to report telemetry. The sidecar has local caching such that a large percentage of precondition checks can be performed from the cache. Additionally, the sidecar buffers outgoing telemetry such that it only calls Mixer infrequently.

Metrics

Default Metrics

To open Prometheus dashboard, enter the following command:

```
kubectl port-forward -n istio-system $(kubectl get pods --namespace istio-system --
↪ selector=app=prometheus --output=jsonpath="{.items..metadata.name}") 9090
```

Open <http://localhost:9090>.

To open Grafana, enter the following command:

```
kubectl port-forward -n istio-system $(kubectl get pods --namespace istio-system --
↪ selector=app=grafana --output=jsonpath="{.items..metadata.name}") 3000
```

Open <http://localhost:3000>.

Tracing

Zipkin vs. Jaeger

6.6.4 Knative

Installation on Minikube

See chapter [Minikube](#) for instructions.

Installation on Azure

See chapter [Azure Kubernetes Service](#) for instructions.

Source-to-Image workflow (only Knative Build)

This is currently the only usage of Knative for MICO.

[Knative Build](#) is used to achieve a *Source-to-Image workflow*. [Knative Serving](#) is currently not used.

Istio is not required to run Knative Build.

Installation

Installing Knative:

```
# Install Knative Build and its dependencies:
kubectl apply --filename https://github.com/knative/build/releases/download/v0.3.0/
↪release.yaml

# Monitor the Knative components:
kubectl get pods --namespace knative-build
```

Authentication to a container registry

Different destinations:

- Docker Hub: `docker.io/ustmico/<image-name>`
- ACR: `ustmicoregistry.azurecr.io/<image-name>`
- GCR: `gcr.io/iaas/<image-name>`

Authentication:

Depending on your target container registry, there are different ways to authenticate.

- Docker Hub: [Basic authentication \(Docker\)](#)
- ACR: [Azure Container Registry authentication with service principals](#)
- GCR: [GoogleCloudPlatform/knative-build-tutorials/docker-build/](#)

DockerHub

Create a new namespace `mico-build-bot` (if not yet created):

```
kubectl create namespace mica-build-bot
```

Create a Secret with the name `build-bot-dockerhub-secret`:

```
apiVersion: v1
kind: Secret
type: kubernetes.io/basic-auth
metadata:
  name: build-bot-dockerhub-secret
  namespace: mica-build-bot
  annotations:
    build.knative.dev/docker-0: https://index.docker.io/v1/
data:
  # Use 'echo -n "username" | base64 -w 0' to generate this string
  username: ${DOCKERHUB_USERNAME_BASE64}
  # Use 'echo -n "password" | base64 -w 0' to generate this string
  password: ${DOCKERHUB_PASSWORD_BASE64}
```

Create a new Service Account manifest which is used to link the build process to the secret. Save this file as `build-bot-dockerhub-service-account.yaml`:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: build-bot-dockerhub
  namespace: mica-build-bot
secrets:
- name: build-bot-dockerhub-secret
```

Provide your DockerHub credentials in Base64 encoding as environment variables:

```
export DOCKERHUB_USERNAME_BASE64=$(echo -n "username" | base64 -w 0)
export DOCKERHUB_PASSWORD_BASE64=$(echo -n "password" | base64 -w 0)
```

Insert the environment variables into the secret configuration file:

```
envsubst < build-bot-dockerhub-secret.yaml > build-bot-dockerhub-secret-with-
→credentials.yaml
```

Apply the manifest files to your cluster:

```
kubectl apply -f build-bot-dockerhub-secret-with-credentials.yaml build-bot-dockerhub-
→service-account.yaml
```

Azure Container Registry

The Authentication to ACR doesn't work yet on Azure!

```
failed to push to destination: unexpected end of JSON input
```

ACR is officially not supported by Kaniko: [Issue#425](#)

Therefore DockerHub ist used.

There are already at least two Service Principals for following requirements:

- Read access from the AKS cluster to pull images
- Write access from the Jenkins Pipeline to push images

Now it's required to create another Service Principal with write access to the registry.

Create a Service Principal:

```
#!/bin/bash
SERVICE_PRINCIPAL_NAME=ust-mico-acr-knative-build

ACR_REGISTRY_ID=$(az acr show --name $ACR_NAME --query id --output tsv)

SP_PASSWD=$(az ad sp create-for-rbac --name $SERVICE_PRINCIPAL_NAME --scopes $ACR_
→REGISTRY_ID --role contributor --query password --output tsv)
SP_APP_ID=$(az ad sp show --id http://$SERVICE_PRINCIPAL_NAME --query appId --output
→tsv)

echo "Service principal ID: $SP_APP_ID"
echo "Service principal password: $SP_PASSWD"
```

Create a new namespace mico-build-bot:

```
kubectl create namespace mico-build-bot
```

Create a Secret with the name build-bot-acr-secret:

```
kubectl create secret docker-registry build-bot-acr-secret --docker-server $ACR_NAME.
→azurecr.io --docker-username $SP_APP_ID --docker-password $SP_PASSWD --
→namespace=mico-build-bot
```

Create a ServiceAccount with the name build-bot-acr:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: build-bot-acr
  namespace: mico-build-bot
secrets:
- name: build-bot-acr-secret
```

Usage of Knative Build

Now we are ready to create and run a Build.

How a simple Build can be created and be executed is written down in [Creating a simple Knative Build](#).

More Build examples: [Knative Build resources](#).

There is already a set of curated and supported Build Templates available in the Knative `build-templates` repository. For a Dockerfile build and the subsequent push of the resulting image the usage of *Kaniko* as the Builder is recommended. It is also used in the *Source-to-URL workflow*.

Example application:

Apply at first a `Secret` and a `Service Account` so that the resulting image can be pushed to a docker registry (here: Docker Hub).

build.yaml:

```
apiVersion: build.knative.dev/v1alpha1
kind: Build
metadata:
  name: build-hello
  namespace: mico-build-bot
spec:
  serviceAccountName: build-bot-dockerhub
  source:
    git:
      url: https://github.com/dgageot/hello.git
      revision: master
  steps:
  - name: build-and-push
    image: gcr.io/kaniko-project/executor
    args:
      - --dockerfile=/workspace/Dockerfile
      - --destination=docker.io/ustmico/hello
```

Increase log level: Add `--verbosity=debug` to args.

Deploying an application (only Knative Serving)

Knative Serving is currently not used in MICO!

There are multiple sample applications, developed on Knative, available: [Knative serving sample applications](#)

Here we use the [Hello World - Spring Boot Java sample](#)...

The app is already available as a Docker image. If you just want to deploy it to the Kubernetes Cluster, you can skip the next section.

Building the app locally

Create a new empty web project:

```
curl https://start.spring.io/starter.zip \
  -d dependencies=web \
  -d name=helloworld \
  -d artifactId=helloworld \
  -o helloworld.zip
unzip helloworld.zip
```

Run the application locally:

```
./mvnw package && java -jar target/helloworld-0.0.1-SNAPSHOT.jar
```

Go to `localhost:8080` to see your Hello World! message.

Create Dockerfile:

```
FROM maven:3.5-jdk-8-alpine as build
ADD pom.xml ./pom.xml
ADD src ./src
RUN mvn package -DskipTests

FROM openjdk:8-jre-alpine
COPY --from=build /target/helloworld-*.jar /helloworld.jar
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/helloworld.jar"]
```

Create service.yaml:

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: helloworld-java
  namespace: default
spec:
  runLatest:
    configuration:
      revisionTemplate:
        spec:
          container:
            image: docker.io/ustmico/knative-helloworld-java
            env:
              - name: TARGET
                value: "Spring Boot Sample v1"
```

Building and deploying the sample with Docker:

```
# Build the container on your local machine
docker build -t ustmico/knative-helloworld-java .

# Push the container to docker registry
docker push ustmico/knative-helloworld-java
```

Deployment to Kubernetes

Deploy the app into the Kubernetes cluster:

```
# Apply the manifest
kubectl apply --filename service.yaml

# Watch the pods for build and serving
kubectl get pods --watch
```

How to access the service is described in section [Access a service](#).

Source-to-URL workflow (Knative Build + Serving)

Knative Serving is currently not used in MICO!

This section describes how to use Knative to go from source code in a git repository to a running application with a URL. Knative Build and Knative Serving are required.

As a sample application we use a Go application like describes in [Orchestrating a source-to-URL deployment on Kubernetes](#).

Install the kaniko build template

This sample leverages the kaniko build template to perform a source-to-container build on your Kubernetes cluster.

```
kubectl apply --filename https://raw.githubusercontent.com/knative/build-templates/  
↪master/kaniko/kaniko.yaml
```

Register secrets for Docker Hub

In order to push the container that is built from source to Docker Hub, register a secret in Kubernetes for authentication with Docker Hub.

Save this file as `docker-secret.yaml` with your Docker Hub credentials:

```
apiVersion: v1  
kind: Secret  
metadata:  
  name: basic-user-pass  
  annotations:  
    build.knative.dev/docker-0: https://index.docker.io/v1/  
type: kubernetes.io/basic-auth  
data:  
  # Use 'echo -n "username" | base64 -w 0' to generate this string  
  username: BASE64_ENCODED_USERNAME  
  # Use 'echo -n "password" | base64 -w 0' to generate this string  
  password: BASE64_ENCODED_PASSWORD
```

Create a new Service Account manifest which is used to link the build process to the secret. Save this file as `service-account.yaml`:

```
apiVersion: v1  
kind: ServiceAccount  
metadata:  
  name: build-bot  
secrets:  
- name: basic-user-pass
```

Apply the manifest files to your cluster:

```
$ kubectl apply --filename docker-secret.yaml  
secret "basic-user-pass" created  
$ kubectl apply --filename service-account.yaml  
serviceaccount "build-bot" created
```

Deploying the sample

You need to create a service manifest which defines the service to deploy, including where the source code is and which build-template to use. Create a file named `service.yaml` and copy the following definition. Make sure to replace `{DOCKER_USERNAME}` with your own Docker Hub username:

```
apiVersion: serving.knative.dev/v1alpha1  
kind: Service  
metadata:  
  name: app-from-source
```

(continues on next page)

(continued from previous page)

```

namespace: default
spec:
  runLatest:
    configuration:
      build:
        apiVersion: build.knative.dev/v1alpha1
        kind: Build
        spec:
          serviceAccountName: build-bot
          source:
            git:
              url: https://github.com/mchmarny/simple-app.git
              revision: master
            template:
              name: kaniko
              arguments:
                - name: IMAGE
                  value: docker.io/{DOCKER_USERNAME}/app-from-source:latest
          revisionTemplate:
            spec:
              container:
                image: docker.io/{DOCKER_USERNAME}/app-from-source:latest
                imagePullPolicy: Always
                env:
                  - name: SIMPLE_MSG
                    value: "Hello sample app!"

```

Deploying the sample to Kubernetes

Deploy the app into the Kubernetes cluster:

```

# Apply the manifest
kubectl apply --filename service.yaml

# Watch the pods for build and serving
kubectl get pods --watch

# Check the state of the service and get the service object
kubectl get ksvc app-from-source --output yaml

```

Access a service

View the IP address of the service:

```
kubectl get svc knative-ingressgateway --namespace istio-system
```

In the following commands replace `<service-name>` with your specified service name (e.g. `helloworld-java` or `app-from-source`).

To find the URL for your service, use

```
kubectl get ksvc <service-name> \
  --output=custom-columns=NAME:.metadata.name,DOMAIN:.status.domain
```

Export the Ingress hostname and IP address as environment variables:

```
export SERVICE_HOST=`kubectl get route <service-name> --output jsonpath="{.status.
↪domain}"`
export SERVICE_IP=`kubectl get svc knative-ingressgateway --namespace istio-system \
--output jsonpath="{.status.loadBalancer.ingress[*].ip}"`
```

Make a request to the app:

```
curl --header "Host:$SERVICE_HOST" http://${SERVICE_IP}
```

Explore the configuration

View the created Route resource:

```
kubectl get route --output yaml
```

View the created Configuration resource:

```
kubectl get configurations --output yaml
```

View the Revision that was created by our Configuration:

```
kubectl get revisions --output yaml
```

Clean up

To clean up a service:

```
kubectl delete --filename service.yaml
```

Monitoring

Accessing logs

See [Accessing logs](#) for more information.

To open the Kibana UI (the visualization tool for Elasticsearch), start a local proxy with the following command:

```
kubectl proxy
```

This command starts a local proxy of Kibana on port 8001. For security reasons, the Kibana UI is exposed only within the cluster. Navigate to the Kibana UI. It might take a couple of minutes for the proxy to work.

Metrics

See [Accessing metrics](#) for more information.

To open Grafana, enter the following command:

```
kubectl port-forward --namespace knative-monitoring $(kubectl get pods --namespace_
↪knative-monitoring --selector=app=grafana --output=jsonpath="{.items..metadata.name}
↪") 3000
```

This starts a local proxy of Grafana on port 3000. For security reasons, the Grafana UI is exposed only within the cluster. Navigate to the Grafana UI at localhost:3000.

Memory usage

Memory usage of an one-node-cluster (total max: 1,7 GB):

- kube-system (total: 315 MB)
 - heapster: 22 MB
 - kube-dns-v20 (1): 21 MB
 - kube-dns-v20 (2): 21 MB
 - kube-proxy: 26 MB
 - kube-svc-redirect: 35 MB
 - kubernetes-dashboard: 13 MB
 - metrics-server: 13 MB
 - omsagent-mvrrx: 102 MB
 - omsagent-rs: 52 MB
 - tunnelfront: 10 MB
- istio-system (total: 456 MB)
 - istio-telemetry: 182 MB
 - istio-policy: 138 MB
 - istio-egressgateway: 29 MB
 - istio-ingressgateway: 26 MB
 - istio-pilot: 45 MB
 - istio-citadel: 11 MB
 - istio-galley: 10 MB
 - istio-sidecar-injector: 9 MB
 - istio-statsd-prom-bridge: 6 MB
- knative-build (total: 22 MB)
 - build-controller: 11 MB
 - build-webhook: 11 MB
- other processes running in cluster: 1 GB

Troubleshooting

Get logs of a build:

Check the Init Container (e.g. pod-name = kaniko-build-f9x52):

```
kubect1 describe pod <pod-name>
```

Accessing logs from Init Containers (e.g. `init-container=build-step-build-and-push`):

```
kubectl logs <pod-name> -c <init-container>
```

6.6.5 Helm

Currently we are only using Helm for templating. Therefore only the client is required (Tiller as the server-side component is not required).

Install Helm client

There are several possibilities [how to install Helm](#). One of them is by using a script:

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get > get_helm.sh
chmod 700 get_helm.sh
./get_helm.sh
```

To initialize Helm without installing Tiller on your cluster, use

```
helm init --client-only
```

Templating with Helm

1. Fetching the chart templates (`helm fetch`)
2. Rendering the template with configurable values (`helm template`)
3. Applying the result to the cluster (`kubectl apply`)

Istio

Helm is great for templating. Especially for creating an adapted Istio installation. The official Istio [installation steps](#) also recommends the usage of using `helm template`.

For more information see section [Istio](#).

Setup Helm & Tiller

Like already mentioned, Tiller is currently not used!

(<https://docs.microsoft.com/en-us/azure/aks/kubernetes-helm>)

- Install Helm on your local machine (<https://github.com/helm/helm#install>)
- Create a service account and an according role-binding for Tiller (necessary in a RBAC-enabled cluster).

```
helm-rbac.yaml:
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
```

(continues on next page)

(continued from previous page)

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: tiller
  namespace: kube-system

```

- Apply it to Kubernetes with:

```
kubectl apply -f helm-rbac.yaml
```

- Alternatively via command-line:

```

kubectl create serviceaccount --namespace kube-system tiller \
kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin \
↪--serviceaccount=kube-system:tiller \
helm init --service-account tiller --upgrade

```

Troubleshooting

- Getting error *Error: found in requirements.yaml, but missing in charts/ directory: ...*

Update dependencies (Istio in version 1.1.0-snapshot.4 is used):

```

helm init --client-only
helm repo add istio.io https://storage.googleapis.com/istio-release/releases/1.1.
↪0-snapshot.4/charts
helm dependency update install/kubernetes/helm/istio

```

6.6.6 Monitoring

Installation

We use the YAML files of the GitHub repository [prometheus/prometheus](#) and from [giantswarm/prometheus](#) for the installation of Prometheus modified for our use.

Install it:

```
kubectl apply -f install/kubernetes/monitoring
```

Prometheus

To open Prometheus dashboard, enter the following command:

```

kubectl port-forward -n monitoring $(kubectl get pods --namespace monitoring --
↪selector="app=prometheus,component=core" --output=jsonpath="{.items..metadata.name}
↪") 9090:9090

```

API

Prometheus HTTP API Documentation

Example 1:

```
curl -g 'http://localhost:9090/api/v1/query?query=container_cpu_load_average_10s'
↳ {container_name="mico-core"}
```

Example 2:

```
curl -g 'http://localhost:9090/api/v1/query?query=container_memory_working_set_bytes'
↳ {pod_name="POD_NAME"}
```

This query uses the metric `container_memory_working_set_bytes` that provides the memory usage. It is limited to the memory usage of the pod with a specified name. Prometheus provides three different values for this request:

- total memory usage of a pod
- memory usage of a container running in the pod
- difference between total memory usage of the pod and the memory usage the containers

Example 3:

```
curl -g 'http://localhost:9090/api/v1/query?query=sum(container_memory_working_set_bytes{pod_name="POD_NAME", container_name=""})'
```

This query adds the requirement `container_name=""`. This filters the three values from example 2. Only the total memory usage value of the pod has an empty container name (because it is a whole pod, not just the container inside). The sum function removes metadata provided together with the metric itself.

Further information

kube-state-metrics

GitHub

Download release: [Release v1.5.0](#)

Install it:

```
kubectl apply -f kubernetes
```

6.6.7 OpenFaaS

Installation

We use the YAML files of the GitHub repository [openfaas/faas-netes](#) for the installation of OpenFaaS modified for our use with the monitoring of other components.

Install it:

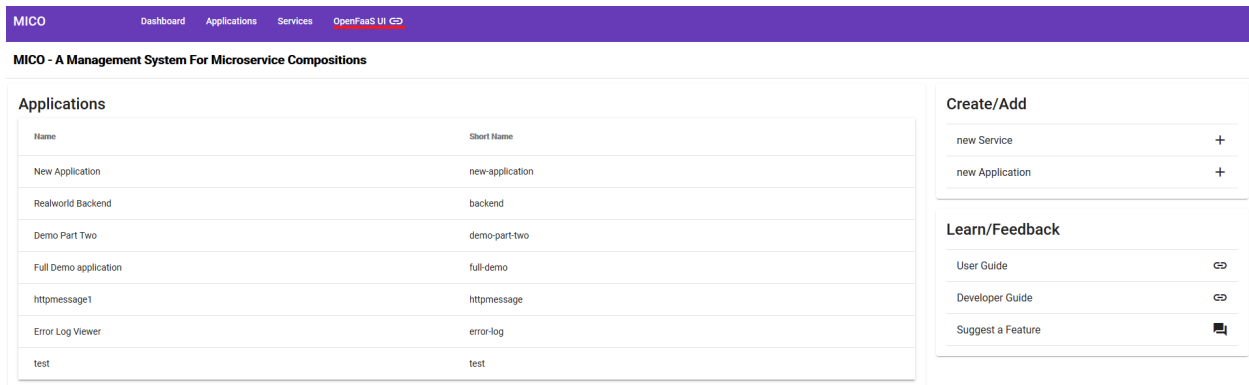
```
kubectl apply -f install/kubernetes/openfaas
```

Function deployment

The fastest method to deploy a function is to use the OpenFaaS Function Store via the OpenFaaS UI. To deploy a custom function the OpenFaaS CLI is necessary. Both methods are described in the following sections.

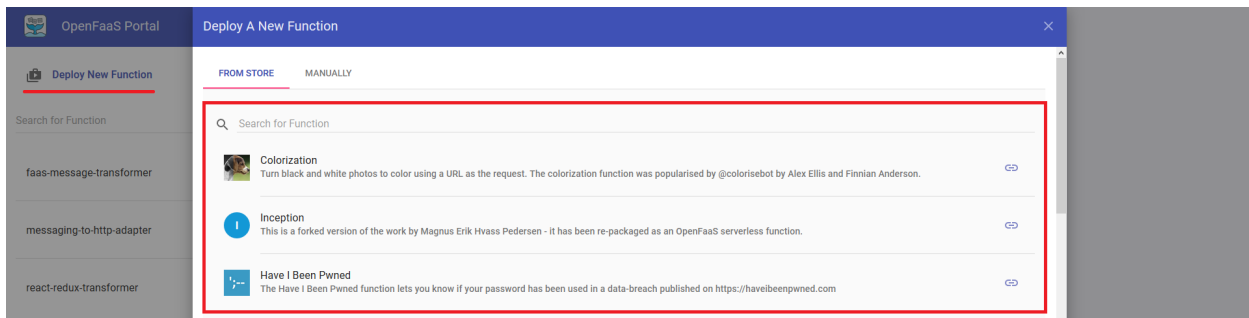
Function Store

1. To open the OpenFaaS UI use the button provided in the MICO UI.



The necessary credentials are configured in the [setup script](#). The default username is admin.

1. After the login is completed the UI offers the ‘Deploy A New Function’ button. The button opens a list and it is possible to select a function from the store.



1. After selecting a function, use the ‘Deploy’ button on the bottom of the list.

Custom Function

To build and deploy a custom function the following requirements are necessary:

- Kubectl must be configured (See the previous sections)
- Docker Hub account
- Docker

This tutorial is tested on windows. It should work on Linux too. All commands are tested in Powershell but should also work in the Git Bash.

1. Install OpenFaaS CLI:
 - (a) Download the FaaS CLI from [here](#)

- (b) (Optional) Point the path variable in windows to the directory of the faas-cli.exe
 - (c) In case of a problem read the readme from the download page.
2. Configure the Docker Hub account in Docker (Docker Icon in Taskbar):
 - (a) Right-click docker icon in the taskbar
 - (b) Select Sign in
 - (c) Enter credentials
3. Start the port forwarding with `kubectkl kubectkl port-forward svc/gateway -n openfaas 8080`
4. Log in with the faas CLI `faas-cli login --password PASSWORD`
5. `faas-cli list` should now provide a list of all deploy functions
6. Change into the directory of a function (The root of a function contains the file `stack.yml` or `<function-Name>.yml`)
7. Edit the yaml file (Not always necessary):
 - (a) The attributes `provider` and `gateway` should have the values `openfaas` and `http://127.0.0.1:8080` (Different values are possible for the gateway if for example no port forwarding is used)
 - (b) The attribute `image` should have the prefix of the Docker Hub account/organization name (e.g. “ust-mico”)
8. Use `faas-cli up` (up = build, push, deploy) if the yaml file has the name `stack.yml` or `faas-cli up -f function-name.yml` if the yaml file has a custom name. (e.g. “faas-cli up -f messaging-to-http-adapter.yml”)
9. Use the function (Look up the URL and other parameters in the OpenFaaS UI)

Troubleshooting

- If the deployment fails, check of the port forwarding is still running. It tends to break often.
- If the build fails, check if all files have Linux line endings (including the `stack.yml`)
- If the push fails, check if the DockerHub account is configured correctly
- If something else breaks read the OpenFaaS documentation or tutorials.

Example functions are located in our custom functions [repository](#)

Function Development

The development of functions is different for each programming language. But it is recommended to use a template for the language from [here](#). The templates are also accessible from the OpenFaaS CLI with `faas-cli new` (e.g. `faas-cli new <function-name> --lang java8`)

Further information

6.6.8 Kafka

Installation

To install Kafka in our cluster we use the setup provided by the community GitHub repository [Yolean/kubernetes-kafka](#).

Modifications:

- kafka/50kafka.yml, zookeeper/50pzoo.yml and zookeeper/51zoo.yml: Remove storageClassName: standard

Install Kafka with its dependencies:

```
kubectl apply -f 00-namespace.yml && kubectl apply -k ./variants/dev-small/
```

Access to Kafka

Kafka is available under `bootstrap.kafka:9092` inside the cluster.

Access to the Kafka Broker:

```
KAFKA_BROKER=$(kubectl get pods -l=app=kafka -n kafka -o jsonpath="{.items[*].  
↪metadata.name}")  
kubectl -n kafka exec -it $KAFKA_BROKER /bin/bash
```

Relevant commands (run inside of a Kafka broker):

List topics:

```
/opt/kafka/bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

List consumer groups:

```
/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
```

Delete topic:

```
/opt/kafka/bin/kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic_  
↪transform-result
```

Consumer:

```
/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --group_  
↪mico --topic transform-result --from-beginning
```

Producer:

```
/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic_  
↪transform-request
```

Sample Cloud Events messages are documented [here](#).

Example:

```
{ "specversion": "0.2", "type": "io.github.ust.mico.result", "source": "/router", "id": "A234-  
↪1234-1234", "time": "2019-05-08T17:31:00Z", "contenttype": "application/json", "data": {  
↪ "key": "value" } }
```

kafkacat

kafkacat is a command line utility that you can use to test and debug Apache Kafka® deployments. You can use kafkacat to produce, consume, and list topic and partition information for Kafka. Described as “netcat for Kafka”, it is a swiss-army knife of tools for inspecting and creating data in Kafka.

It is similar to Kafka Console Producer (`kafka-console-producer`) and Kafka Console Consumer (`kafka-console-consumer`), but even more powerful.

`kafkacat` is an open-source utility, available at <https://github.com/edenhill/kafkacat>. It is not included in Confluent Platform.

Deploy `kafkacat` to Kubernetes:

```
kubectl apply -f 01-test-namespace.yml && kubectl apply -f ./kafka/test/kafkacat.yml
```

Access to `kafkacat`:

```
KAFKACAT=$(kubectl get pods -n test-kafka -l test-target=kafka-client-kafkacat -  
↪o=jsonpath={.items..metadata.name})  
kubectl -n test-kafka exec -it $KAFKACAT /bin/bash
```

Listing topics:

```
kubectl -n test-kafka exec -it $KAFKACAT -- kafkacat -b bootstrap.kafka:9092 -L
```

Consuming topic 'test-kafkacat':

```
kubectl -n test-kafka exec -it $KAFKACAT -- kafkacat -b bootstrap.kafka:9092 -C -t_  
↪test-kafkacat
```

Producing message to topic 'test-kafkacat':

```
kubectl -n test-kafka exec -it $KAFKACAT -- kafkacat -b bootstrap.kafka:9092 -P -t_  
↪test-kafkacat
```

Consuming topic 'transform-result' with group id 'mico':

```
kubectl -n test-kafka exec -it $KAFKACAT -- kafkacat -b bootstrap.kafka:9092 -G mico -  
↪C -t transform-result
```

Producing message to topic 'transform-request' with group id 'mico':

```
kubectl -n test-kafka exec -it $KAFKACAT -- kafkacat -b bootstrap.kafka:9092 -G mico -  
↪P -t transform-request
```

Kafka Manager

Install Kafka Manager (Yahoo):

```
kubectl apply -f ./yahoo-kafka-manager
```

Access Kafka Manager via port forwarding:

```
kubectl port-forward svc/kafka-manager -n kafka 30969:80
```

Open <http://localhost:30969>.

Configure Kafka Manager so it's able to access ZooKeeper:

1. Add Cluster
2. Cluster Name: MICO-Kafka

3. Cluster Zookeeper Hosts: `zookeeper.kafka:2181`
4. `brokerViewThreadPoolSize`: 2
5. `offsetCacheThreadPoolSize`: 2
6. `kafkaAdminClientThreadPoolSize`: 2

MICO runs and operates on Kubernetes. This chapter describes different hosting environments and relevant third-party components like Istio and Knative.

This chapter covers all information about the current MICO setup. The setup itself is described in *MICO Setup*. The Azure Kubernetes Service (AKS) is the used for the current Kubernetes development environment and is described in *Azure Kubernetes Service*. The Google Kubernetes Engine is also tested with MICO and *Google Kubernetes Engine*.

More details about different Kubernetes components that are used for MICO are described in section *Kubernetes*.

All instructions are meant to be executed with Bash in a Debian/Ubuntu like development environment. If you are using Windows 10, consider to use *Windows Subsystem for Linux (WSL)*. Alternatives could be *Cygwin* or *Git Bash*.

At first *Install and Set Up kubectl*.

Installation on Ubuntu / Debian:

```
sudo apt-get update && sudo apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/
↪sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
```


7.1 Testing

Depending on which component of MICO you want to test, there are different ways to do that. Usually the local testing of the MICO backend or the MICO dashboard with access to the Kubernetes cluster are the preferred ways:

- *Local testing of the MICO backend with access to the cluster*
- *Local testing of the MICO dashboard with access to the cluster*

7.1.1 Local testing without cluster

This is only an option if you don't need the Kubernetes API. Be aware that all calls made by `mico-core` to the Kubernetes API will cause an exception.

Use `docker-compose` to start `mico-core`, `neo4j` and `redis` as the database:

```
docker-compose up --build
```

To only start `neo4j` and `redis` use

```
docker-compose up neo4j redis
```

7.1.2 Local testing of the MICO backend with access to the cluster

With this approach you are able to test and debug `mico-core` locally. Furthermore you are able to connect the Neo4j browser to a local instance (simple way to view what's going on in the database). It is required to have `kubectl` installed on your system. If you don't have `kubectl` installed yet, you should do that now. For more information see [Setup](#).

Start Neo4j and Redis locally. The easiest way is to use `docker-compose`:

```
docker-compose up neo4j redis
```

To start `mico-core` locally there are basically 3 ways:

- Use your IDE (e.g. IntelliJ IDEA), to be able to debug `mico-core` (**recommended**)
- Use maven to build and start `mico-core`:

```
# Build mico-core with maven without executing tests
mvn -B -f mico-core/pom.xml clean package -Dmaven.test.skip=true

# Start mico-core with the profile `dev`
java -jar -Dspring.profiles.active=dev mico-core/target/mico-core-1.0-SNAPSHOT.jar
```

- Use `docker-compose` to start `mico-core` within a Docker container. **This approach is currently not an option** because `kubectl` and a proper Kubernetes configuration is missing in the Docker image.

Per default `mico-core` uses the Spring profile `dev` (set as active in `application.properties`). The `dev` profiles configures `mico-core` to use the Kubernetes namespace `mico-testing` as the target for the Builds and the namespace for the resulting *MicoServices*. This namespace must include the Kubernetes Secret and the ServiceAccount for the authentication to DockerHub. To create the namespace `mico-testing` alongside the required resources, you can use the script `/install/kubernetes/test-setup-only-build-environment.sh`.

Last but not least you should provide a way to connect to Prometheus. This is required if you want to retrieve the status of a deployed `MicoApplication` and its `MicoServices`. The easiest way is connect to the running Prometheus instance inside your cluster by establishing a port-forwarding:

```
kubectl port-forward svc/prometheus -n monitoring 9090:9090
```

Testing mico-core

To make requests to the locally running `mico-core` you can either use `mico-admin` (start it also locally) or make requests with `curl` or *Postman*. Some Postman collections are already provided. See the [Postman section](#) for more information.

Neo4j

You can connect to the remote interface of Neo4j via <http://localhost:7474/browser>. In the Neo4j browser you are able to use the Cypher Shell and execute cypher statements.

Get all:

```
MATCH (n)
RETURN n;
```

Delete all:

```
MATCH (n)
DETACH DELETE n;
```

7.1.3 Local testing of the MICO dashboard with access to the cluster

`mico-admin` needs access to `mico-core`. You have two options:

- Start `mico-core` locally (see [section above](#))
- Access `mico-core` in the Kubernetes cluster by using a port forwarding:

```
kubectl port-forward svc/mico-core -n mico-system 8080:8080
```

Be aware that you make requests to `mico-core` inside of the cluster in the namespace `mico-system`. If you want to test with an own instance of `mico-core`, consider to create an own namespace like described in the following section [Testing in cluster with own namespace](#). Even then you are able to use a local instance of `mico-admin` and access `mico-core` inside of the cluster. If you have choosen e.g. the namespace name `mico-testing-1337` you are able to access `mico-core` by using the port forwarding `kubectl port-forward svc/mico-core -n mico-testing-1337 8080:8080`.

See the [README](#) of `mico-admin` for information how to start it locally.

7.1.4 Testing in cluster with own namespace

Note: *This approach is not practical anymore because of the required resources. Use a different approach.*

To be able to test in an exclusive environment, you can deploy MICO in a separate namespace inside the cluster. For this purpose the script `/install/kubernetes/test-setup-all-in-own-namespace.sh` in the [MICO repository](#) can be used.

Preparation

Prepare yourself to enter a name for the test namespace (e.g. `mico-testing-1337`) as well as your credentials for DockerHub.

Now you are able to execute the interactive setup script `install/kubernetes/test-setup-all-in-own-namespace.sh`.

The Neo4j database needs ~5min until it is ready. You can check the current status with

```
kubectl get pods -n ${NAMESPACE} --watch
```

Usage

`mico-admin` gets after some seconds a public IP. To watch the service for changes use

```
kubectl get svc mico-admin -n ${NAMESPACE} --watch
```

To get the external IP directly use

```
echo $(kubectl get svc mico-admin -n ${NAMESPACE} -o 'jsonpath={.status.loadBalancer.
  ↳ingress[0].ip}')
```

You can use the IP to access the dashboard in your browser.

`mico-core` can't be accessed from the outside. You can either use `kubectl proxy` or `kubectl port-forward` to get access into the cluster:

- `kubectl proxy --port=8080`

Example:

```
curl localhost:8080/api/v1/namespaces/${NAMESPACE}/services/mico-core/proxy/
↪applications
```

- `kubectl port-forward svc/mico-core -n ${NAMESPACE} 8080:8080`

Example:

```
curl localhost:8080/applications
```

If you really want to expose mico-core to the outside, you can change the type of the Kubernetes Service to LoadBalancer:

```
kubectl patch svc mico-core -n ${NAMESPACE} --type='json' -p ' [{"op": "replace", "path":
↪"/spec/type", "value": "LoadBalancer"} ] '
```

Afterwards you can get the external IP of mico-core by using

```
echo $(kubectl get svc mico-core -n ${NAMESPACE} -o 'jsonpath={.status.loadBalancer.
↪ingress[0].ip}')
```

Update Kubernetes deployment in cluster

To update a single deployment you can change the Docker image of the specific deployment. Use this approach only in your separate namespace.

Example with deployment mico-core:

```
# Build a new Docker image (change tag name e.g. to the corresponding issue number) ↵
↪and push it to DockerHub
docker build -f Dockerfile.mico-core -t ustmico/mico-core:42-fix-api . && docker push ↵
↪ustmico/mico-core:42-fix-api

# Set the image of the Kubernetes deployment
kubectl set image deployment/mico-core mico-core=ustmico/mico-core:42-fix-api -n $
↪${NAMESPACE}

# If there was another image set, the pods of the deployment will automatically be ↵
↪restarted with the new image
# If the image was already set, you must restart the pod manually
kubectl -n ${NAMESPACE} delete pod $(kubectl get pods -n ${NAMESPACE} --selector=run=mico-
↪core --output=jsonpath={.items..metadata.name})
```

Clean up

If you are finished with testing, delete the namespace:

```
kubectl delete namespace ${NAMESPACE}
```

7.1.5 Testing in cluster with Telepresence

Telepresence is a tool to debug Kubernetes services locally.

For Windows users: Telepresence basically supports WSL, however some important features are missing (see [Windows support](#)). Therefore I wasn't able to get it running in my environment.

Installation

Installing Telepresence:

```
curl -s https://packagecloud.io/install/repositories/datawireio/telepresence/script.  
→deb.sh | sudo bash  
sudo apt install --no-install-recommends telepresence
```

Usage with local `mico-core`

Build `mico-core` with Docker:

```
docker build -f Dockerfile.mico-core -t ustmico/mico-core:localbuild .
```

Swap the Kubernetes deployment with the local Docker image:

```
telepresence --namespace mico-system --swap-deployment mico-core --docker-run \  
--rm -it ustmico/mico-core:localbuild
```

Usage with local `mico-admin`

Build `mico-admin` with Docker:

```
docker build -f Dockerfile.mico-admin -t ustmico/mico-admin:localbuild .
```

Swap the Kubernetes deployment with the local Docker image:

```
telepresence --namespace mico-system --swap-deployment mico-admin --docker-run \  
--rm -it ustmico/mico-admin:localbuild
```

Usage with IntelliJ

See [Debugging a Java Rate Limiter Service using Telepresence and IntelliJ IDEA](#).

7.1.6 Misc

Check current configuration

`mico-core` offers some additional endpoints that are part of the Spring Boot Actuator plugin.

To get the current configurations have a look at `localhost:8080/actuator/configprops`. To format it and save it into a file use e.g.

```
curl localhost:8080/actuator/configprops | python3 -m json.tool > configprops.json
```

Or as an alternative with `jq`:

```
curl localhost:8080/actuator/configprops | jq . > configprops.json
```

7.2 Debugging

7.2.1 Neo4j

To execute arbitrary queries you can use the Cypher Shell that is included in the Neo4j Docker image.

Access `cypher-shell`:

```
# Get the pod name of the Neo4j deployment (change the namespace if necessary)
NEO4J_POD_NAME=$(kubectl get pods -n mico-system --selector=run=neo4j --
  ↳output=jsonpath={.items..metadata.name})

# Run bash inside the container
kubectl exec $NEO4J_POD_NAME -n mico-system -it /bin/bash

# Run cypher-shell
bash-4.4# cypher-shell
```

Now you are able to execute queries.

Get all nodes:

```
MATCH (n) RETURN n
```

Delete all nodes and relationships:

```
MATCH (n)
DETACH DELETE n
```

For more queries see the official documentation of [Cypher](#).

7.3 Postman

We share our Postman collections and environment configurations via [Git](#).

7.3.1 Install

[Install Postman](#)

7.3.2 Update API

Everyone that changes the API of one of the MICO Microservices should also update the Postman collections and export them to this directory. An easy way to do that is by using the Swagger UI, copy the API docs, and import it into Postman.

Example with `mico-core`:

- Port forwarding to `mico-core` to be able to access the REST API: `kubectl port-forward svc/mico-core -n mico-system 8080:8080`
- Open <http://localhost:8080/v2/api-docs> (Swagger-UI would be <http://localhost:8080/swagger-ui.html>)
- Copy the complete JSON
- Postman -> Import

- Optional: Change the property `"host": "localhost:8080"` to `"host": "{{host}}"` so that Postman will use the host definition from the environment configuration
- Export updated collection

This chapter covers how to debug and test MICO.

Tutorials for how to use MICO.

8.1 How To add a service

To add a new service click on the *new Service* button in the dashboard or the service overview.

MICO

DashboardApplicationsServices

MICO - A Management System For Microservice Compositions

Applications

| Name | Short Name |
|------|------------|
|------|------------|

Create/Add

new Service

+

new Application

+

Learn/Feedback

User Guide

↔

Developer Guide

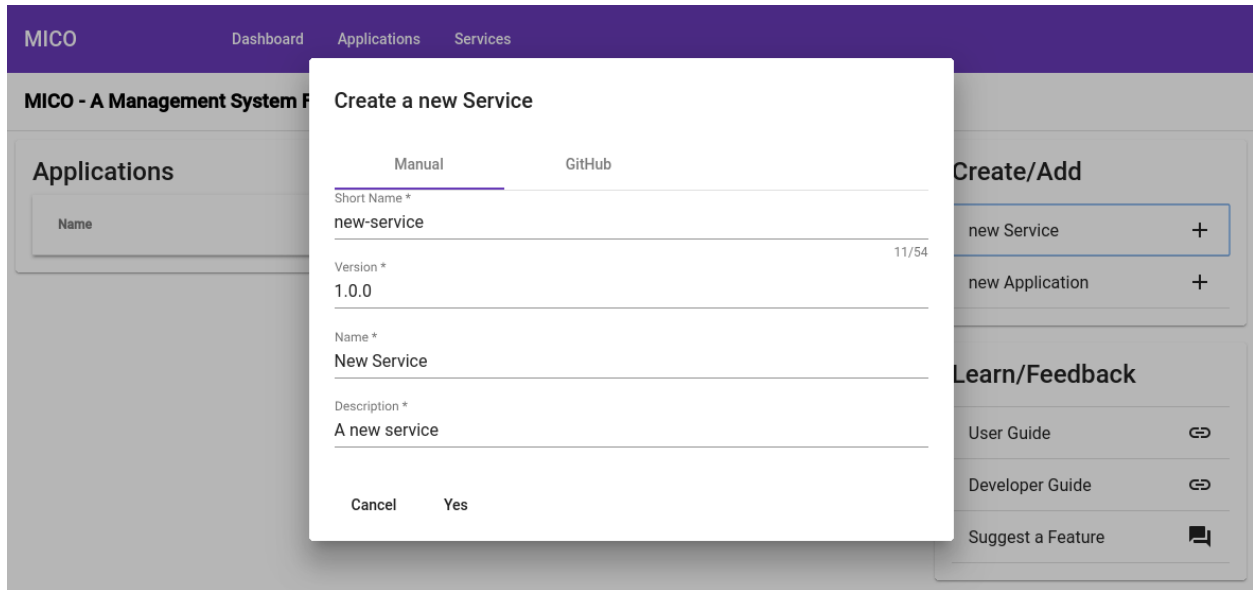
↔

Suggest a Feature

🗨

8.1.1 Manual service creation

To create a service manually fill in all the fields in the dialog:



Note: `Short Name` must be all lowercase, contain only (latin) letters, numbers and the character `-`. Also the first character must be a letter.

`Version` must consist of three numbers formatted like in <https://semver.org/spec/v2.0.0.html> (semantic versioning).

8.1.2 Importing a service from GitHub

Open the *GitHub* tab in the dialog and enter the url to the github repository.

Hint: The [UST-MICO Organization](#) on GitHub has two example repositories that you can import directly.

- [spring-boot-realworld-example-app](#)
 - [react-redux-realworld-example-app](#)
-

In the next step you can choose to import the *latest* or a specific version.

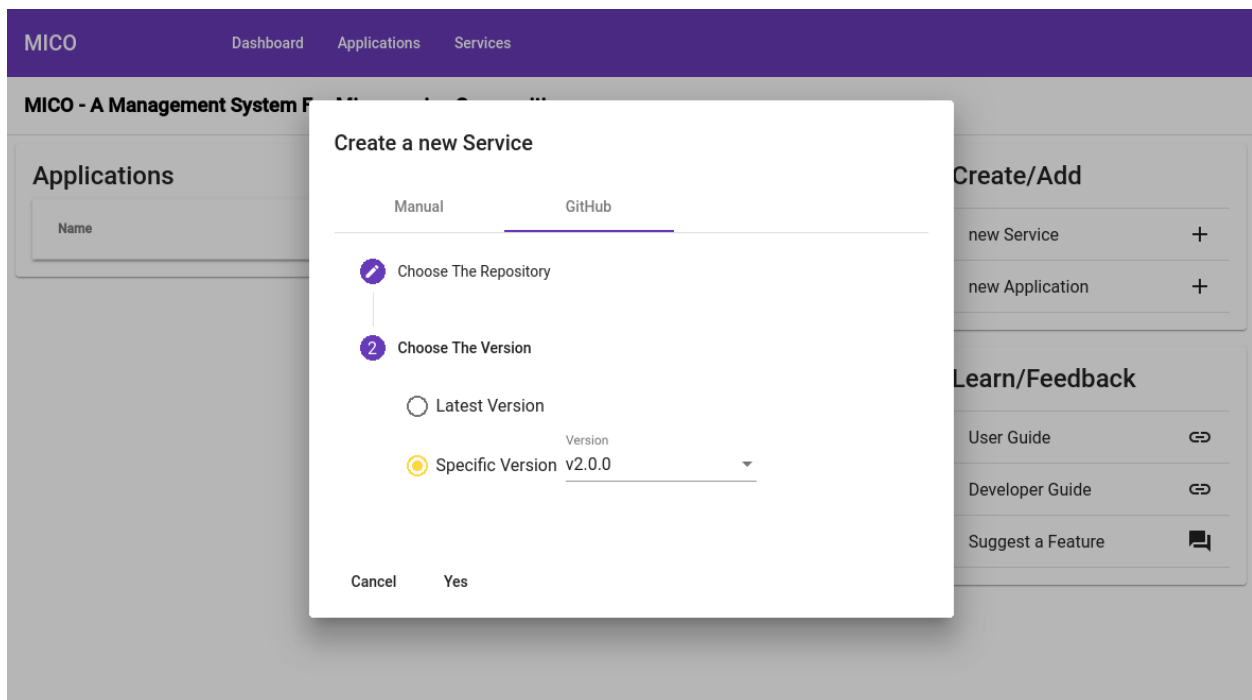
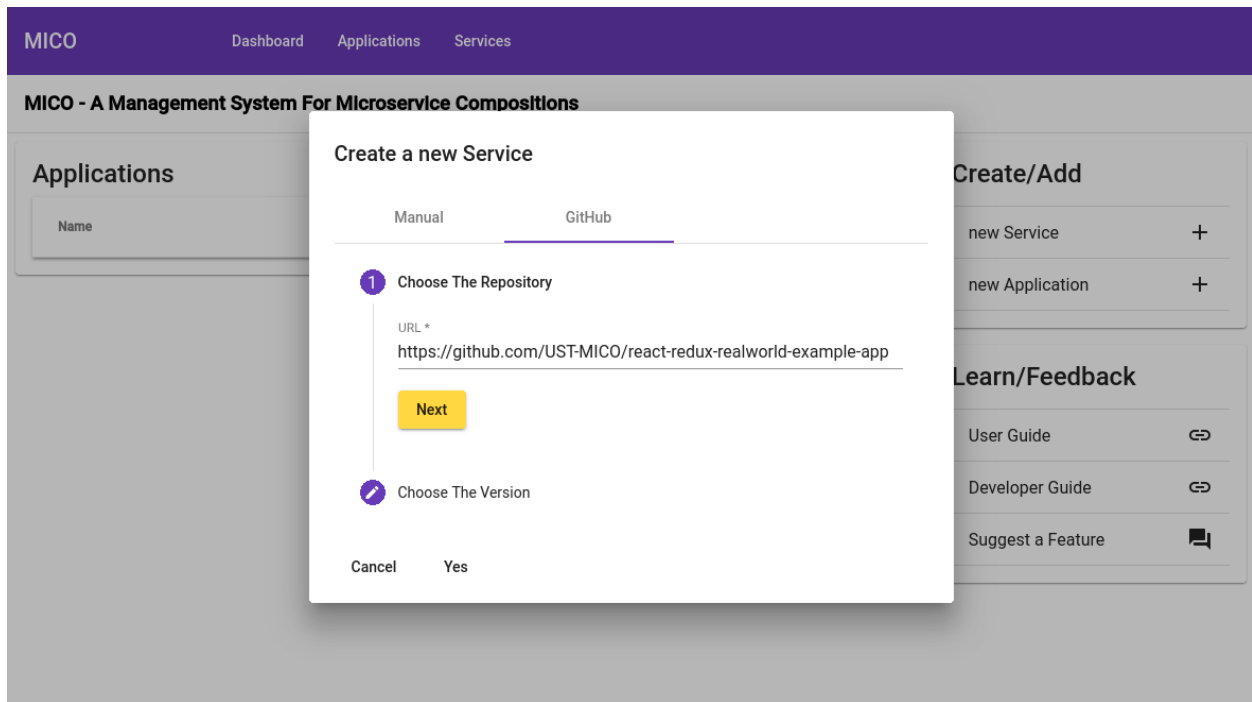
Warning: The import from GitHub feature only works if the repository has releases that match the [semantic versioning](#) schema! It is also necessary to include a `dockerfile` in the root folder of the repository.

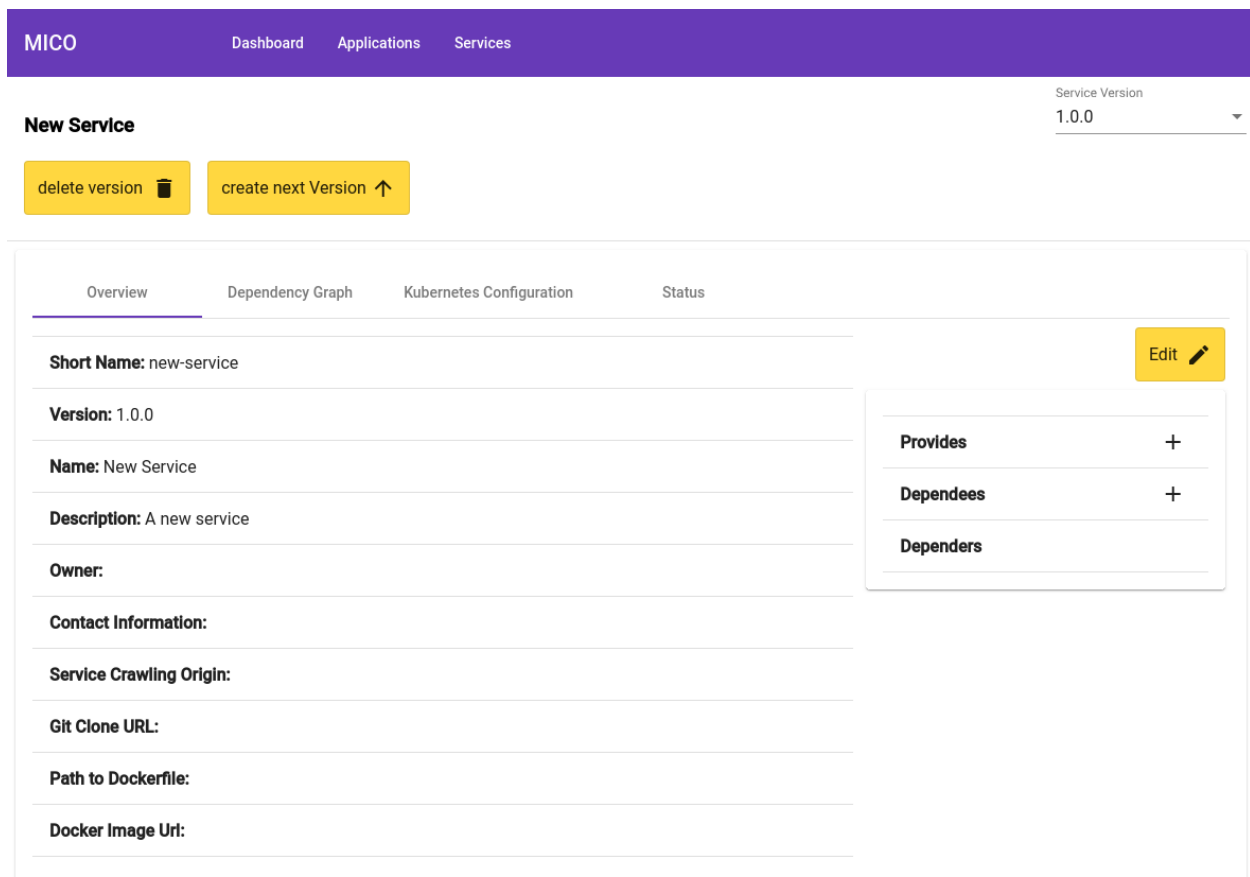
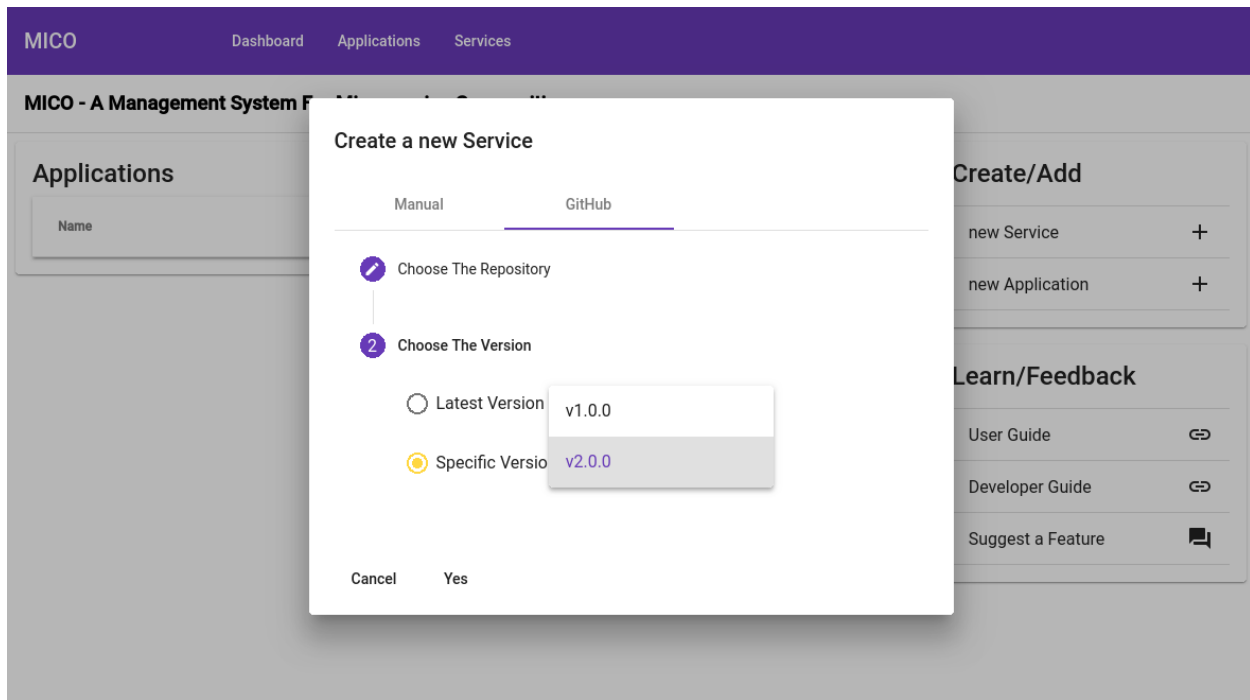
You can also choose a specific release to import.

8.1.3 After adding the service

You will be redirected to the service detail page, if the service was added successfully.

In the top right corner you can select the services version which is displayed.





Right below the service name on the left, you can delete this specific service version with the *delete version* button. Further, you can create the next service version, if the currently selected version is the latest. Thereby, the next version number (next major, next minor, next patch) can be selected. The new version will be a copy of the previous latest version with an updated version number.

On the right side you can find a button to *Edit* the service and Buttons to add *service interfaces Provides* and to add dependencies *Dependees* to this service.

MICO
Dashboard
Applications
Services

New Service
Service Version
1.0.0

delete version
create next Version

Overview
Dependency Graph
Kubernetes Configuration
Status

Short Name *
new-service

Version *
1.0.0

Name *
New Service

Description *
A new service

Owner

Contact Information

Git Clone URL

Path to Dockerfile

Docker Image Url

Save
Cancel

Edit

Provides
Dependees
Dependers

8.2 How To manage a service

8.2.1 Manage service interfaces


The *service interfaces* are used to map the open ports of the service docker container to ports of the *Kubernetes service*. To add a new service interface click on the *Provides* button in the *service detail* page.


MICO

DashboardApplicationsServices

New Service

Service Version
1.0.0

delete version 

create next Version 

OverviewDependency GraphKubernetes ConfigurationStatus

Short Name: new-service

Version: 1.0.0

Name: New Service

Description: A new service

Owner:


Contact Information:

Service Crawling Origin:

Git Clone URL:

Path to Dockerfile:

Docker Image Uri:

Edit 

Provides +

Dependees +

Dependers

The screenshot shows the MICO web application interface. At the top is a purple navigation bar with the MICO logo and links to Dashboard, Applications, and Services. The main content area is titled 'New Service' and contains a sidebar with service details and a main form area. A modal dialog titled 'Create a new Service Interface' is open in the center.

Service Details (Sidebar):

- Short Name: new-service
- Version: 1.0.0
- Name: New Service
- Description: A new service
- Owner:
- Contact Information:
- Service Crawling Origin:
- Git Clone URL:
- Path to Dockerfile:
- Docker Image Uri:

Create a new Service Interface Modal:

- Service Interface Name ***: interface (9/54)
- Description**: (empty)
- Protocol**: (empty)
- Ports**: current: 1, min: 1, max: ∞
 - Ports 1** (minus button):
 - Exposed Port Number ***: 80 (Value min: 1, Value max: 65535)
 - Target Port Number ***: 8080 (Value min: 1, Value max: 65535)
 - Type ***: TCP (dropdown)
 - (plus button)
- Buttons**: Cancel, Create

Warning: At least one port must be provided per service interface!

Note: Exposed Port Number is the port that will be exposed in kubernetes.

Target Port Number is the port exposed by the docker container.

Note: For this tutorial the following assumptions are made:

The service UST-MICO/react-redux-realworld-example-app has an interface frontend with Exposed Port Number 80, Target Port Number 80 and type TCP.

The service UST-MICO/spring-boot-realworld-example-app has an interface backend with Exposed Port Number 80, Target Port Number 8080 and type TCP.

After adding a interface to a service, the detail page should look something like this:

MICO Dashboard Applications Services

New Service Service Version 1.0.0

delete version create next Version

Overview Dependency Graph Kubernetes Configuration Status

Short Name: new-service

Version: 1.0.0

Name: New Service

Description: A new service

Owner:

Contact Information:

Service Crawling Origin:

Git Clone URL:

Path to Dockerfile:

Docker Image Uri:

Edit

Provides +

| Name | Interface | Protocol | Ports |
|-----------|-----------|---------------------|-------|
| interface | TCP | 80 (internal: 8080) | |

Dependees +

Dependers

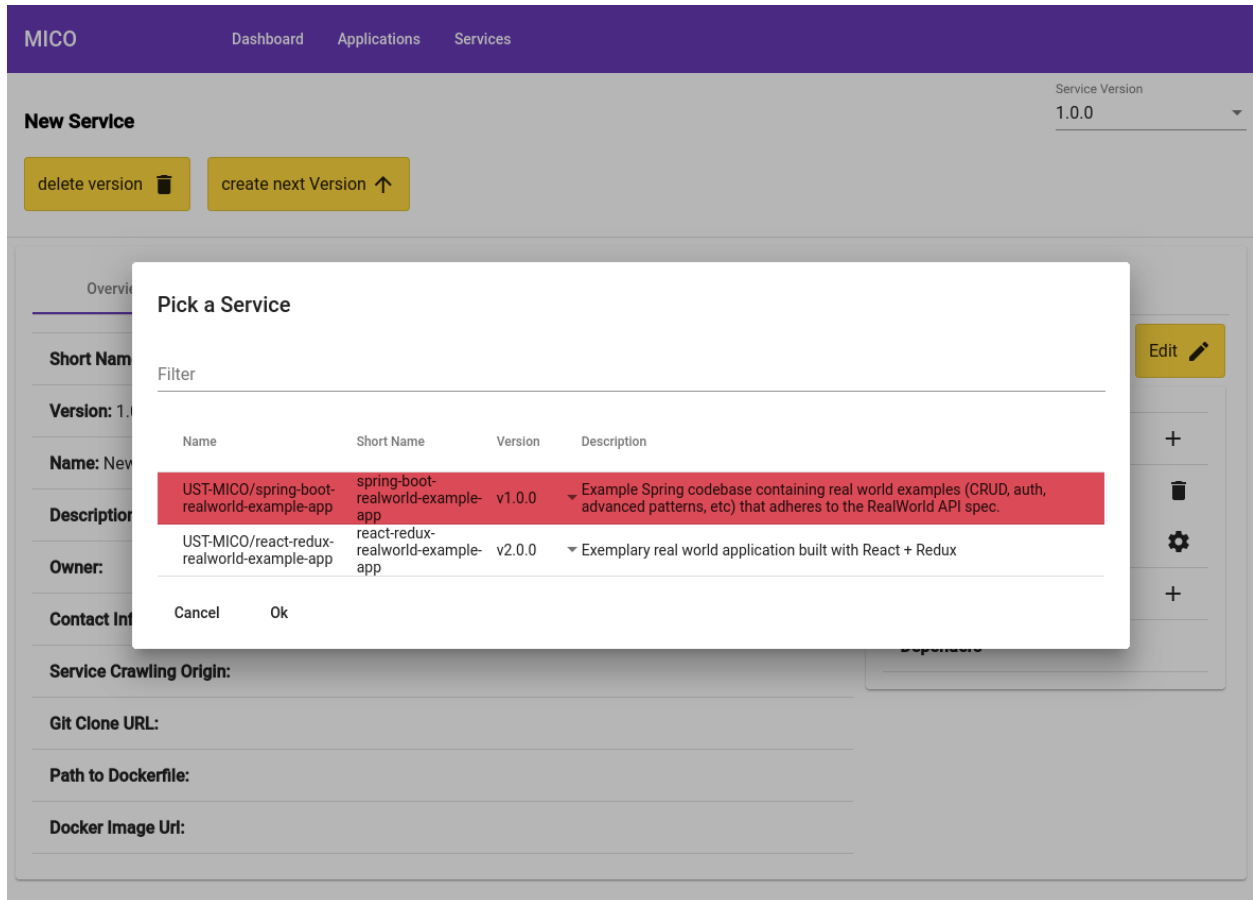
To edit a service interface, click on the text of the service interface.

To delete a service interface, use the trashbin icon right beside the service interface.

8.2.2 Manage service dependencies

To add an existing service as a dependency to this service, use the *Dependees* button.

You can then select the service (and the specific version of that service) that you want to have as a dependency.



If you have added a dependency, the name of the dependency will show up below the *Dependees* button.

You can directly go to the detail page of a dependency by clicking on the name of the dependency.

To remove a dependency, use the trashbin icon right beside its name.

You can also view the whole dependency graph for the service in the *Dependency Graph* tab.

In the dependency graph you can directly change the version of a dependency if the dependency is directly connected to this service (the blue service node). To change the version of a dependency click on its version number and choose a different version in the dialog.

Warning: Changing the dependency graph may break existing applications that use this service.

8.3 How To add an application

First *add some services* to your MICO instance.

MICO Dashboard Applications Services

New Service Service Version 1.0.0

delete version create next Version

Overview Dependency Graph Kubernetes Configuration Status

Short Name: new-service

Version: 1.0.0

Name: New Service

Description: A new service

Owner:

Contact Information:

Service Crawling Origin:

Git Clone URL:

Path to Dockerfile:

Docker Image Uri:

Edit

Provides

Name: interface Protocol:

Transport Protocol:

Ports: TCP 80 (Internal: 8080)

Dependees

UST-MICO/spring-boot-real...

Dependers

MICO Dashboard Applications Services

New Service Service Version 1.0.0

delete version create next Version

Overview Dependency Graph Kubernetes Configuration Status

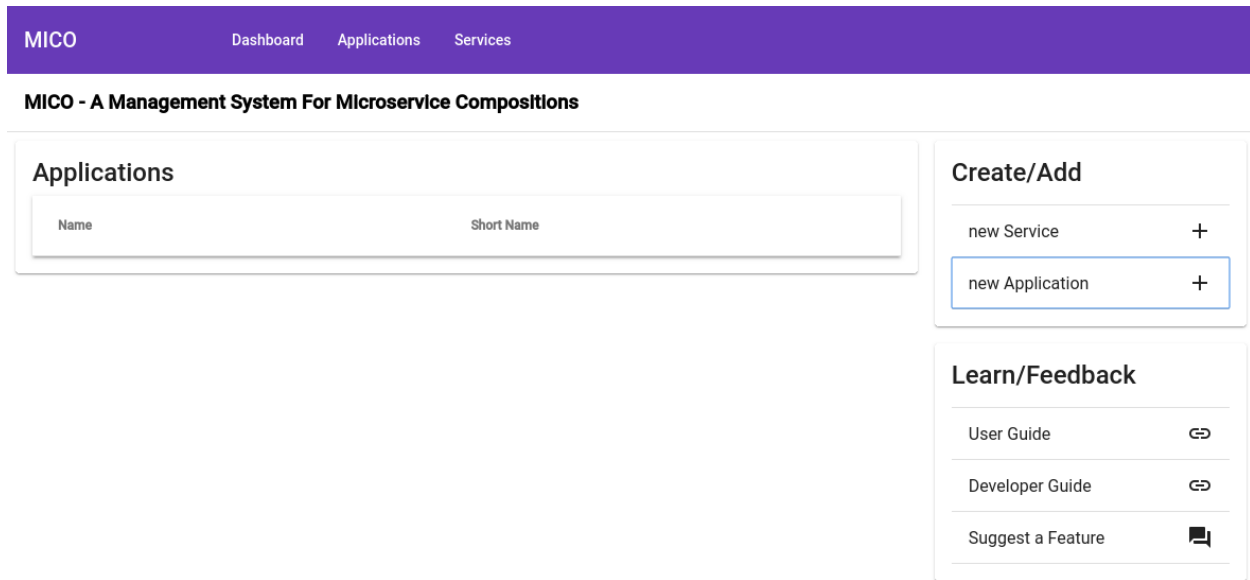
Auto-zoom

New Service
A new service
1.0.0

UST-MICO/sprin...
Example Spring
codebase containing
real world examples (C...
v1.0.0

UST-MICO/react-...
Exemplary real world
application built with
React + Redux
v1.0.0

To add a new application click on the *new Application* button in the dashboard or the application overview.



Fill in the values of the dialog. For `Short Name` and `Version` the same restrictions as for services apply.

You can choose the services for the new application now with the *pick Services* button.

Services still can be added, removed or changed after the creation.

Pick the services with their specific version. Only one version of a service may be directly included in an application. This does not restrict dependencies of included services.

The selected services are shown in the create new application dialog.

8.3.1 After adding the application

You will be redirected to the application detail page, if the application was added successfully.

On the application detail page, you can choose the version of the application in the top right corner.

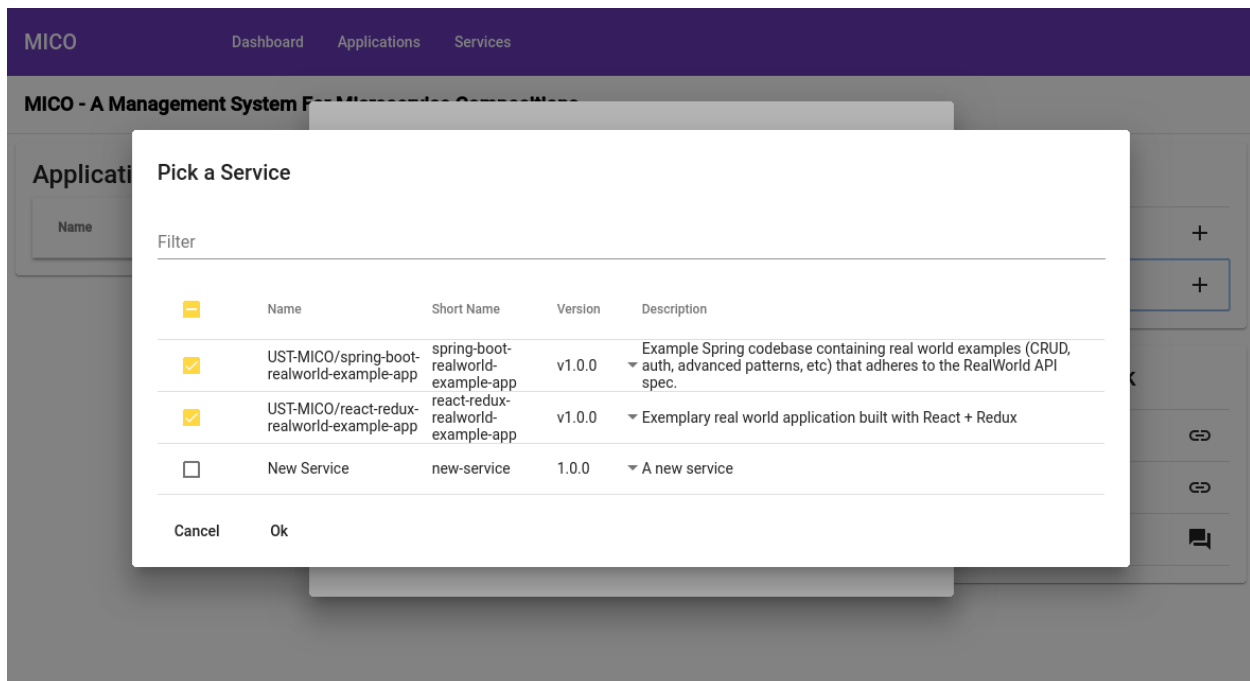
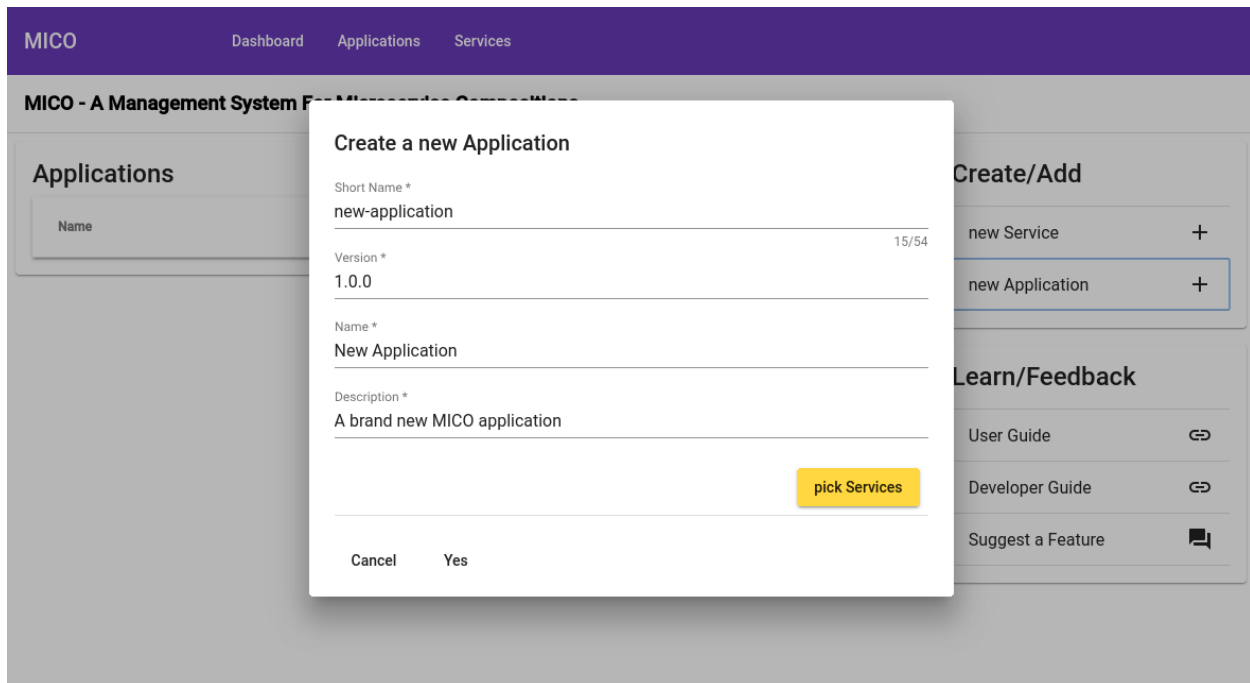
Right below the application name on the left, you can deploy the application with the *deploy* button, undeploy a the application with the *undeploy* button, create a new version of this application with the *create next Version* button or delete this specific application version with the *delete version* button.

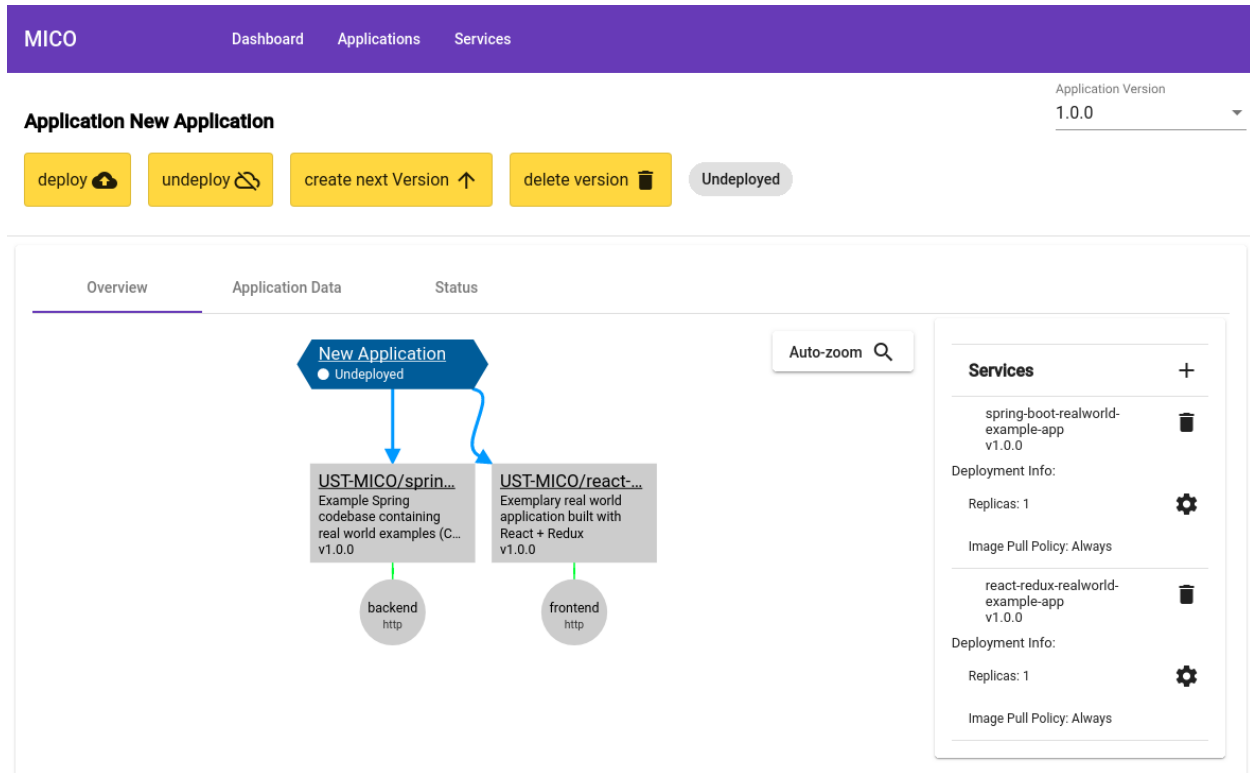
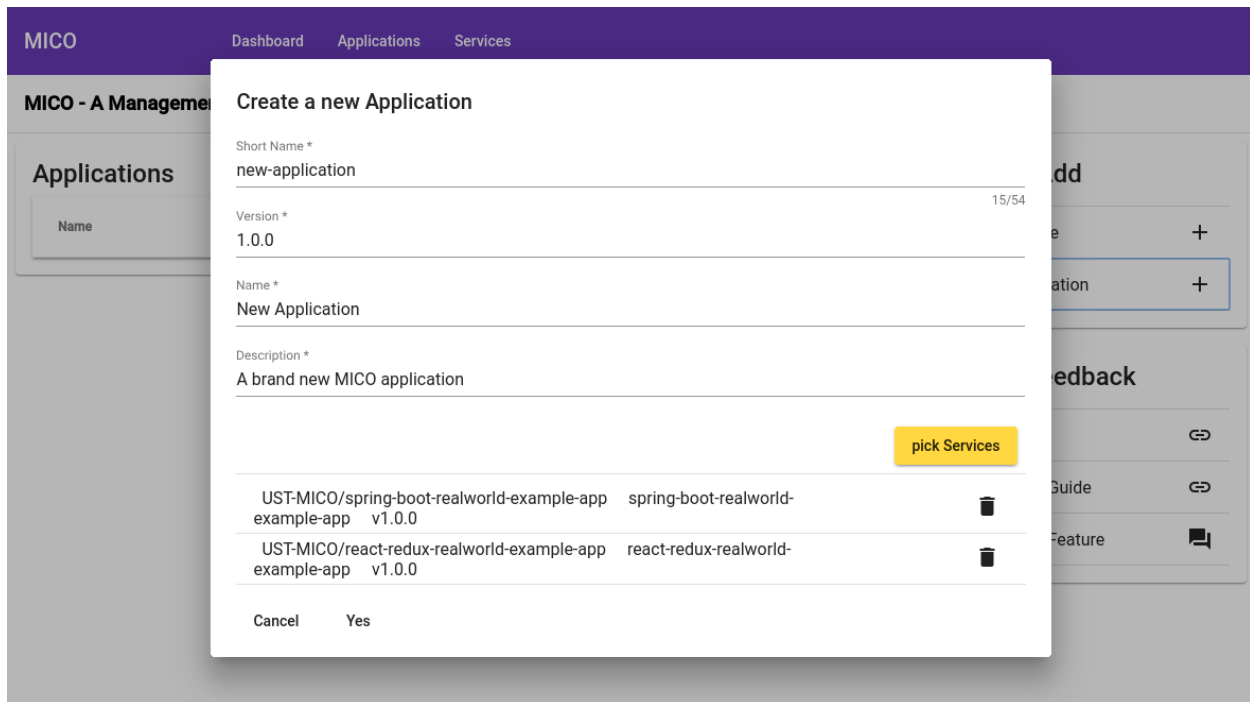
On the right side you can find a *Services* button to add services to the application. The included services are listed below this button together with their deployment information.

To edit the application data switch to the tab *Application Data*.

8.4 How To manage an application

First *add some services* and an *application* to your MICO instance.





MICO

DashboardApplicationsServices

Application New Application

Application Version
1.0.0

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

Edit

Short Name: new-application

Version: 1.0.0

Name: New Application

Description: A brand new MICO application

Owner:

Contact:

Deployment Status

Holds the current deployment status of this application.

messages

Element 1

content: MicoApplication is currently not deployed.

type: Info

value: Undeployed

Note: For this tutorial the following assumptions are made:

The services `UST-MICO/react-redux-realworld-example-app` and `UST-MICO/spring-boot-realworld-example-app` have been *imported from github*.

The service `UST-MICO/react-redux-realworld-example-app` has an interface frontend with Exposed Port Number 80, Target Port Number 80 and type TCP.

The service `UST-MICO/spring-boot-realworld-example-app` has an interface backend with Exposed Port Number 80, Target Port Number 8080 and type TCP.

The Application New Application includes both services.

To add the services to the application, use the *Services* button on the right.

The screenshot shows the MICO dashboard with a purple header bar containing 'MICO', 'Dashboard', 'Applications', and 'Services'. Below the header, there's a section titled 'Application New Application' with buttons for 'deploy', 'undeploy', 'create next Version', 'delete version', and 'Undeployed'. The 'Application Version' is set to '1.0.0'. The main area has tabs for 'Overview', 'Application Data', and 'Status'. The 'Overview' tab is active, showing a diagram of a 'New Application' (Undeployed) with two services: 'UST-MICO/spring-boot-realworld-example-app' (backend http) and 'UST-MICO/react-redux-realworld-example-app' (frontend http). On the right, a 'Services' panel lists the two services with their deployment info: 'Replicas: 1' and 'Image Pull Policy: Always'.

8.4.1 Editing deployment information of services

Some services need environment variables at runtime to work correctly. This is also true for the `UST-MICO/react-redux-realworld-example-app` service.

The service needs the environment variable `BACKEND_REST_API` to *find the backend at runtime*.

The deployment info for a service contains these environment variables and other deployment specific information, like the number of replicas, to start. It is displayed below the service in the box on the right.

To edit the deployment information use the gear icon.

In the dialog you can edit the following settings:

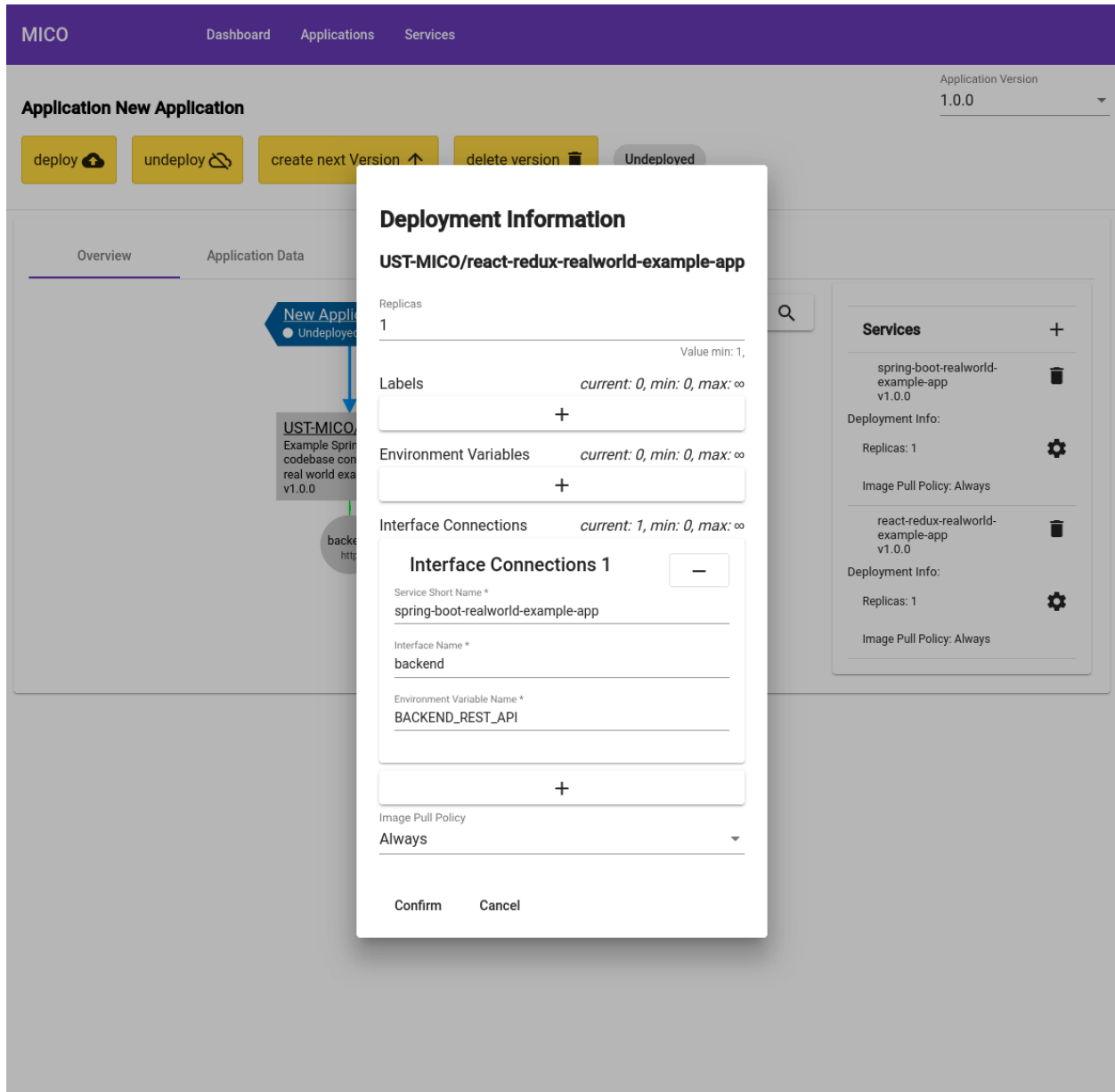
Replicas The number of replicas to scale the service.

Labels Key value pairs that get mapped to kubernetes labels.

Environment Variables Environment variables to set for the container.

Interface Connections Connections to service interfaces of other services. The address of the service interface gets stored in a specified environment variable.

Image Pull Policy When to reload image from dockerhub.



To connect the frontend with the backend service, we need to add an interface connection with the following values to the frontend service:

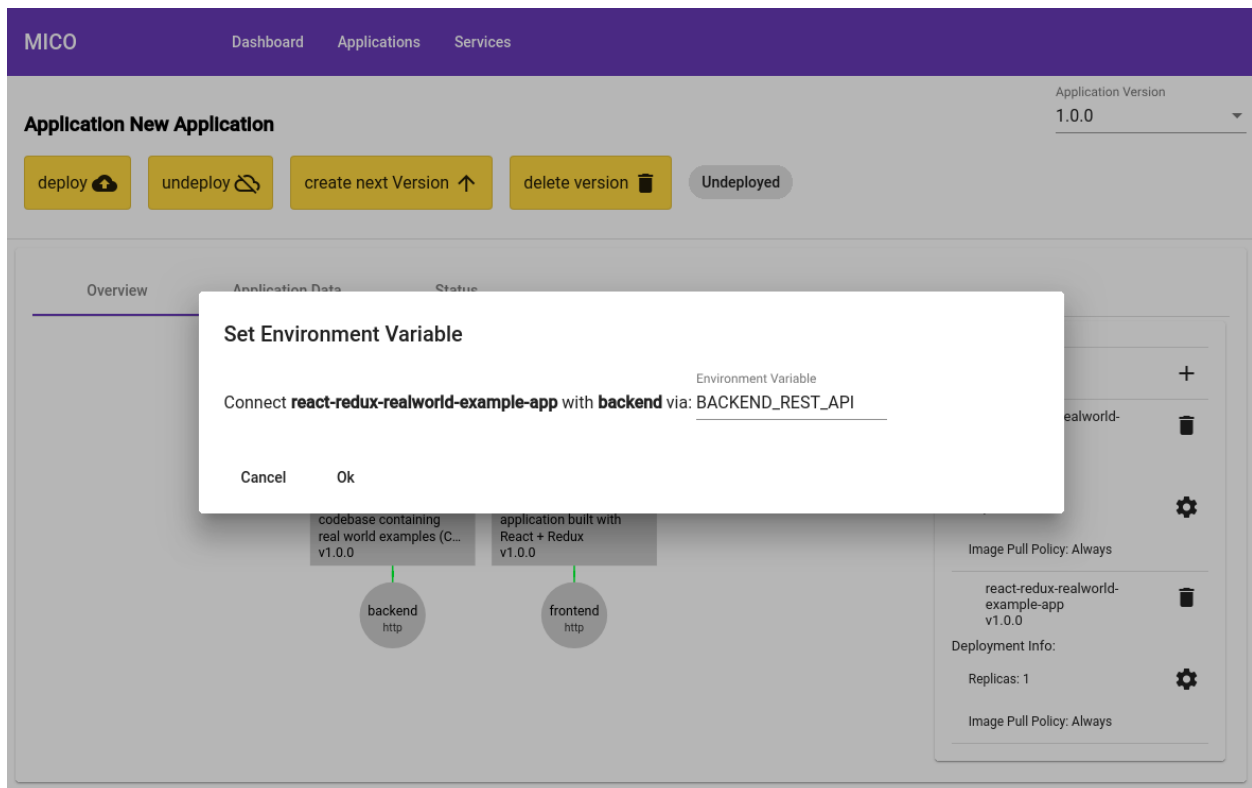
- **Environment Variable Name:** BACKEND_REST_API
- **Interface Name:** backend

- **Service Short Name:** `spring-boot-realworld-example-app`

Warning: The actual values depend on the included services of the application!
The assumptions for this tutorial are listed in the note at the top of this tutorial.

You can also connect the service to the backend interface in the grapheditor. To do this hover with the mouse over the `UST-MICO/react-redux-realworld-example-app` service in the graph and drag one of the appearing circles to the backend interface.

After dropping the edge on the backend interface insert the `BACKEND_REST_API` into the dialog and confirm.



8.4.2 Deploying an Application

To deploy an application, use the *deploy* button below the application name. The actual deployment can take a few minutes, depending on the services included in the application.

Upon completion of the deployment, you can see a list of ips under *Public IPs*. Each service has its own ip.

Note: *kubectl* commands to see what's going on in the background (also helpful if you need troubleshooting):

- Watch all pods in the whole cluster: `kubectl get pods --all-namespaces --watch`
- Watch the pods that are building the Docker images in the background: `kubectl -n mico-build-bot get pods --watch`
- Get the builds: `kubectl -n mico-build-bot get builds`

MICO

DashboardApplicationsServices

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Deployed

OverviewApplication DataStatus

New Application
Deployed

UST-MICO/spring...
Example Spring
codebase containing
real world examples (C...
v1.0.0

UST-MICO/react-f...
Exemplary real world
application built with
React + Redux
v1.0.0

backend
http

frontend
http

Auto-zoom

Services

spring-boot-realworld-
example-app
v1.0.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

react-redux-realworld-
example-app
v1.0.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

Public IPs:
backend: [35.242.245.187](#)
frontend: [35.234.117.92](#)

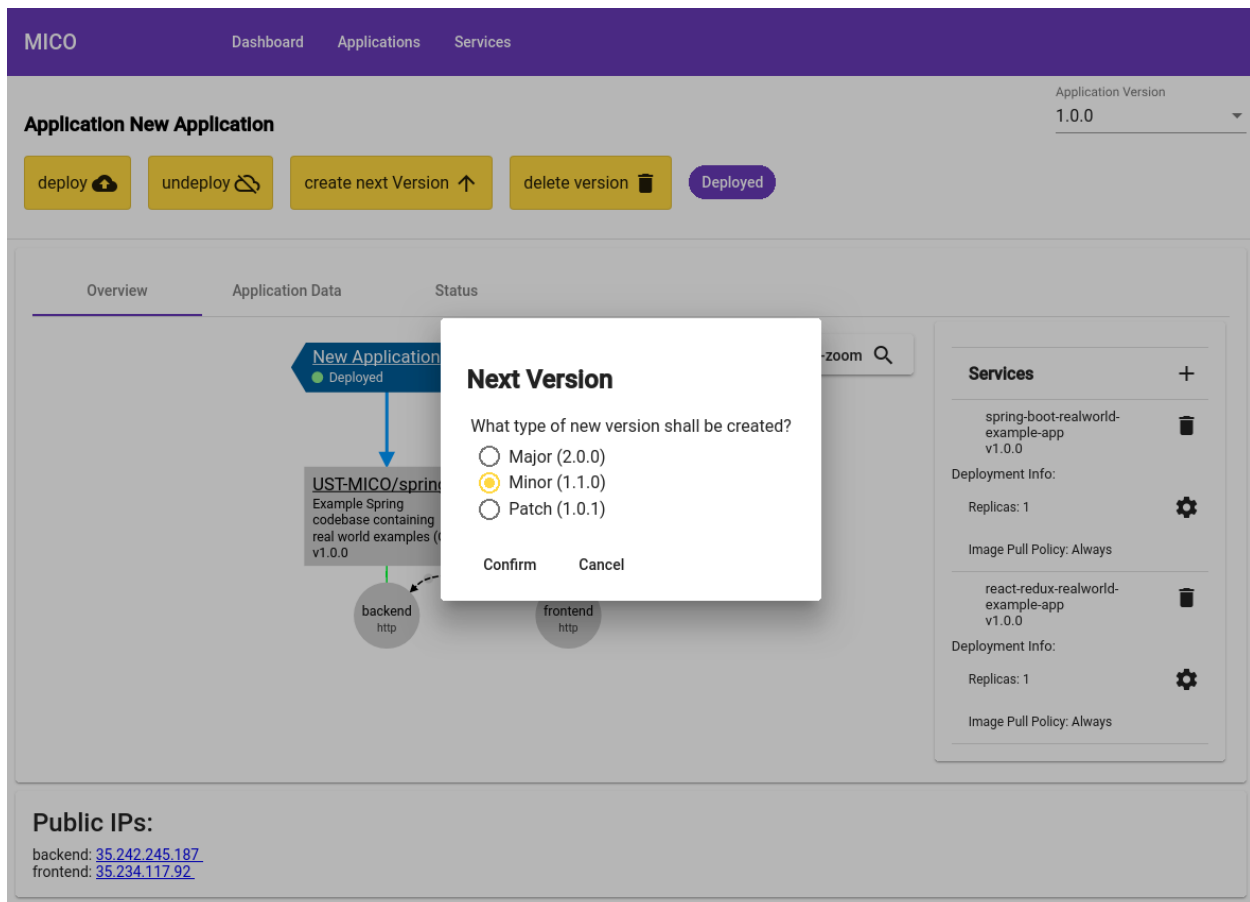
- Get information about a specific build: `kubectl -n mico-build-bot describe build build-react-redux-realworld-example-app-v1-0-0`
- Get the logs of a specific build: `kubectl -n mico-build-bot logs <pod-name> -c build-step-build-and-push`
- Get all resources that were created as part of the deployment of the application(s): `kubectl -n mico-workspace get all`

8.4.3 Undeploying an Application

To undeploy a deployed application, use the *undeploy* button below the application name.

8.4.4 Create a new Application Version

To create a new application version, use the *create next Version* Button and choose the next version to create.



The new version of the application will have all services and deployment information of the old version.

8.5 How To create a messaging based application

MICO supports two different communication types between microservices. The standard communication type is `http`. These connections use the *service interfaces* as described in *How To manage an application*. The second communication type is *cloud events* over `Kafka`.

8.5.1 Kafka enabled services

To use the provided `Kafka` broker a service has to be marked as `Kafka Enabled`. Such a service does not need to specify an `http interface` to be deployable.

See *How To add a service* for details on how to add and edit a service.

Note: The [UST-MICO Organization](#) on GitHub has three example repositories containing `Kafka` enabled services that you can import directly.

- [UST-MICO/test_message_generator](#)
 - [UST-MICO/msg-validator](#)
 - [UST-MICO/http-to-messaging-adapter](#)
-

Hint: For all `Kafka` enabled services the following environment variables will be set:

KAFKA_BOOTSTRAP_SERVERS The URLs of the bootstrap servers

KAFKA_GROUP_ID The group id to be used when subscribing to a `Kafka` topic

KAFKA_TOPIC_INPUT The topic to receive messages from

KAFKA_TOPIC_OUTPUT The topic to send messages to

KAFKA_TOPIC_INVALID_MESSAGE The topic, to which invalid messages are forwarded

KAFKA_TOPIC_DEAD_LETTER The topic, to which messages are forwarded that can't/shouldn't be delivered

KAFKA_TOPIC_TEST_MESSAGE_OUTPUT The topic, to which test messages are forwarded at the end of a test

Input and output topic can be set by the user creating the application. The rest are set by the MICO system.

8.5.2 The Kafka-FaaS-Connector

The `Kafka-FaaS-Connector` is a special service that can be used to implement most of the [Enterprise Integration Patterns](#) by Gregor Hohpe and Bobby Woolf. The communication with `Kafka` and some of the *cloud event attributes* are managed by the `Kafka-FaaS-Connector`. Custom behaviour is implemented as a `OpenFaaS-Function`. The function can be managed with the supplied `OpenFaaS` dashboard that is linked to from the MICO dashboard. MICO will import the latest version of the `Kafka-FaaS-Connector` on startup automatically.

See also:

More background information:

- [Kafka as Messaging Middleware](#)
- [FaaS](#)
- [Function to component mapping](#)

- *[WIP] Kafka-faaS-connector requirements (aka generic component requirements)*
- *Simple Composition Components*
- *Implementation of complex EAI-Patterns with FaaS*

8.5.3 Creating messaging based networks

Note: For this tutorial the following assumptions are made:

The services `UST-MICO/test_message_generator` and `UST-MICO/msg-validator` have been *imported from github*.

Both services have to be edited to set the Kafka Enabled bit!

The service `UST-MICO/msg-validator` has an interface `validator` with Exposed Port Number 80, Target Port Number 8090 and type TCP.

The Application New Application includes both services.

Warning: The services have to be marked as Kafka enabled **BEFORE** being imported into the application to get all default settings for Kafka enabled services!

The starting configuration should look like this:

To link a Kafka enabled service with Kafka topic hover over the service and drag one of the appearing link handles.

The dragged edge can be dropped in the empty space to create a new topic.

In the add topic you can specify the topic and the role. The topic can either be used as a input to the service or as an output from the service. A service can only have one input and one output topic in the grapheditor.

After confirming the dialog the topic appears in the grapheditor. If a topic is already in the grapheditor services can also be linked directly to the topics by dragging an edge from the topic to the service or from the service to the topic. The direction of the arrow symbolises the message flow.

Add the topic as input topic to the `UST-MICO/msg-validator` service. This is a minimal configuration for a messaging based application.

To remove a link drage the link from the link handle near the link target on the link.

Drop the link on empty space to delete it.

To add a Kafka-FaaS-Connector to the application simply drag a edge from a topic and drop it over empty space. The Kafka-FaaS-Connector may take a few seconds to appear.

Dragg a edge from the Kafka-FaaS-Connector to connect it with a new topic.

To change the FaaS function of the Kafka-FaaS-Connector click on the title of the Kafka-FaaS-Connector node to open the dialog.

The title will change to the set FaaS function.

MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application

Undeployed

UST-MICO/test_...

A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-v...

Provides an example
component which reads
messages from a Kafka...
v1.0.2

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:
Replicas: 1
Image Pull Policy: Always

148

Chapter 8. MICO Tutorials

MICO

DashboardApplicationsServicesOpenFaaS UI

Application New Application

Application Version1.0.0

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application

Undeployed

UST-MICO/test...

A message producer, that generates messages randomly for testing pur... v1.1.0

UST-MICO/msg-v...

Provides an example component which reads messages from a Kafka... v1.0.2

Auto-zoom

Services

test-message-generator v1.1.0

Deployment Info: Replicas: 1 Image Pull Policy: Always

msg-validator v1.0.2

Deployment Info: Replicas: 1 Image Pull Policy: Always

8.5. How To create a messaging based application

149

MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application
Undeployed

UST-MICO/test...
A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-v...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

Auto-zoom

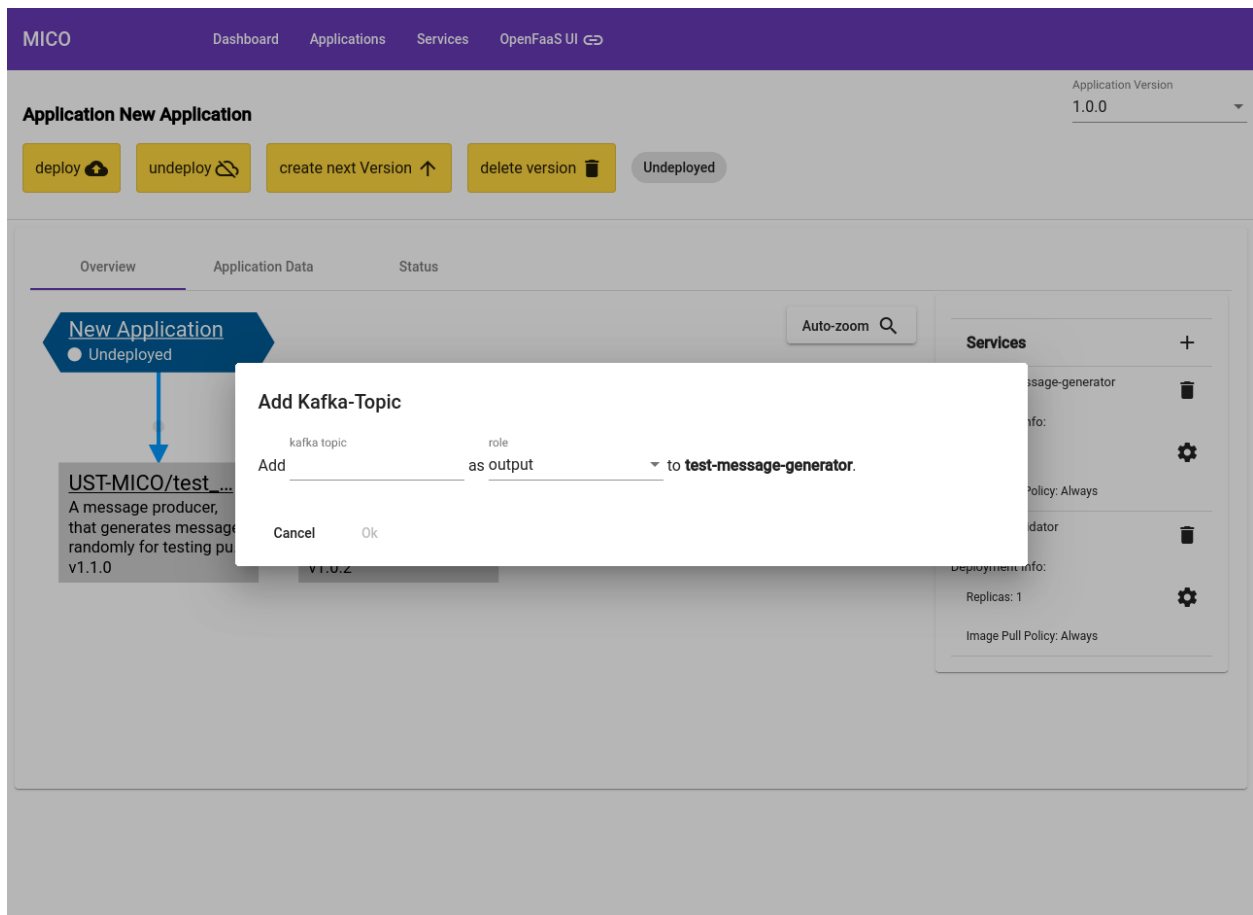
Services

test-message-generator
v1.1.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:
Replicas: 1
Image Pull Policy: Always



MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application
● Undeployed

UST-MICO/test_...
A message producer,
that generates messages
randomly for testing pu...
v1.1.0

UST-MICO/msg-v...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

OUTPUT
test

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:
Replicas: 1
Image Pull Policy: Always

MICO

DashboardApplicationsServicesOpenFaaS UI

Application New Application

Application Version1.0.0

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application

Undeployed

UST-MICO/test_...

A message producer, that generates messages randomly for testing pu... v1.1.0

OUTPUT

test

INPUT

UST-MICO/msg-v...

Provides an example component which reads messages from a Kafka... v1.0.2

Auto-zoom

Services

test-message-generator v1.1.0

Deployment Info: Replicas: 1 Image Pull Policy: Always

msg-validator v1.0.2

Deployment Info: Replicas: 1 Image Pull Policy: Always

8.5. How To create a messaging based application

153

MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application
● Undeployed

UST-MICO/test_...
A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-v...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

test

OUTPUT

INPUT

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:
Replicas: 1
Image Pull Policy: Always

MICO

DashboardApplicationsServicesOpenFaaS UI

Application New Application

Application Version1.0.0

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application

Undeployed

UST-MICO/test...
A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-v...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

OUTPUT

test

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:
Replicas: 1
Image Pull Policy: Always

8.5. How To create a messaging based application

155

MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application
Undeployed

UST-MICO/test-...
A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-v-...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

OUTPUT

test

INPUT

kafka-faa-...
v1.0.1 kafka-...

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:

Replicas: 1

Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:

Replicas: 1

Image Pull Policy: Always

MICO

DashboardApplicationsServicesOpenFaaS UI

Application New Application

Application Version1.0.0

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application

Undeployed

UST-MICO/test...
A message producer, that generates messages randomly for testing pur...
v1.1.0

UST-MICO/msg-V...
Provides an example component which reads messages from a Kafka...
v1.0.2

test

transfo...

kafka-faa...
v1.0.1 kafka-...

OUTPUT

INPUT

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:

Replicas: 1

Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:

Replicas: 1

Image Pull Policy: Always

MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deploy

undeploy

create next Version

delete version

Undeployed

OverviewApplication DataStatus

New Application
● Undeployed

UST-MICO/test...
A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-v...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

test

transfo...

kafka-faa...
v1.0.1 kafka-...

OUTPUT

INPUT

OUTPUT

INPUT

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:

Replicas: 1

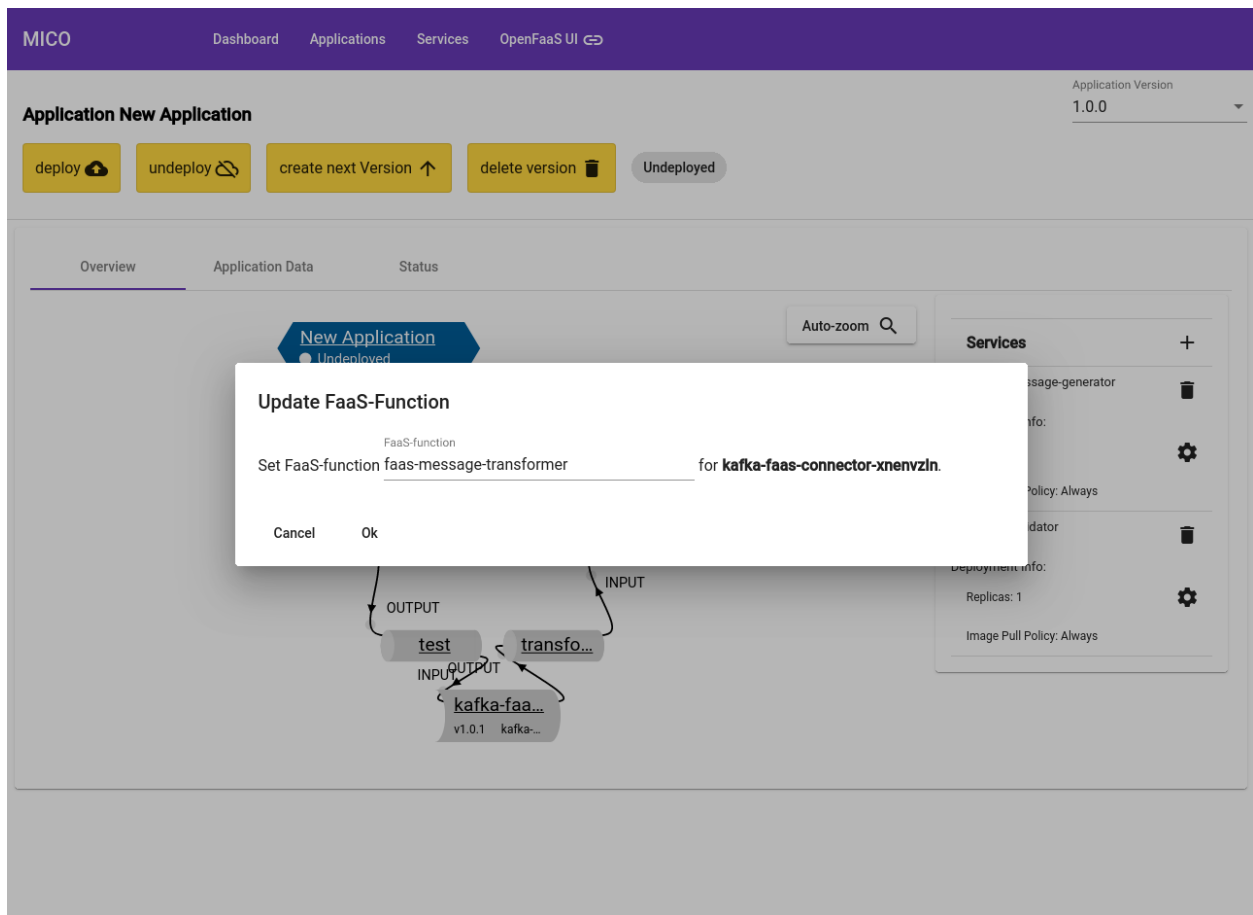
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:

Replicas: 1

Image Pull Policy: Always



MICO

DashboardApplicationsServicesOpenFaaS UI

Application Version
1.0.0

Application New Application

deployundeploycreate next Versiondelete versionUndeployed

OverviewApplication DataStatus

New Application
● Undeployed

UST-MICO/test...
A message producer,
that generates messages
randomly for testing pur...
v1.1.0

UST-MICO/msg-V...
Provides an example
component which reads
messages from a Kafka...
v1.0.2

test

transfo...

faas-mes...
v1.0.1 kafka...

OUTPUT

INPUT

OUTPUT

INPUT

Auto-zoom

Services

test-message-generator
v1.1.0

Deployment Info:
Replicas: 1
Image Pull Policy: Always

msg-validator
v1.0.2

Deployment Info:
Replicas: 1
Image Pull Policy: Always

Architectural Decision Log

This log lists the architectural decisions for MICO.

9.1 [short title of solved problem and solution]

- Status: [accepted | superseded by [ADR-0005](#) | deprecated | ...]
- Deciders: [list everyone involved in the decision]
- Date: [YYYY-MM-DD when the decision was last updated]

Technical Story: [description | ticket/issue URL]

9.1.1 Context and Problem Statement

[Describe the context and problem statement, e.g., in free form using two to three sentences. You may want to articulate the problem in form of a question.]

9.1.2 Decision Drivers

- [driver 1, e.g., a force, facing concern, ...]
- [driver 2, e.g., a force, facing concern, ...]
- ...

9.1.3 Considered Options

- [option 1]
- [option 2]

- [option 3]
- ...

9.1.4 Decision Outcome

Chosen option: “[option 1]”, because [justification. e.g., only option, which meets k.o. criterion decision driver | which resolves force force | ... | comes out best (see below)].

Positive Consequences

- [e.g., improvement of quality attribute satisfaction, follow-up decisions required, ...]
- ...

Negative consequences

- [e.g., compromising quality attribute, follow-up decisions required, ...]
- ...

9.1.5 Pros and Cons of the Options

[option 1]

[example | description | pointer to more information | ...]

- Good, because [argument a]
- Good, because [argument b]
- Bad, because [argument c]
- ...

[option 2]

[example | description | pointer to more information | ...]

- Good, because [argument a]
- Good, because [argument b]
- Bad, because [argument c]
- ...

[option 3]

[example | description | pointer to more information | ...]

- Good, because [argument a]
- Good, because [argument b]
- Bad, because [argument c]

- ...

9.1.6 Links

- [Link type] [Link to ADR]
- ...

9.2 Use Markdown Architectural Decision Records

9.2.1 Context and Problem Statement

We want to record architectural decisions made in this project. Which format and structure should these records follow?

9.2.2 Considered Options

- [MADR 2.1.0](#) - The Markdown Architectural Decision Records
- [Michael Nygard's template](#) - The first incarnation of the term “ADR”
- [Sustainable Architectural Decisions](#) - The Y-Statements
- Other templates listed at https://github.com/joelparkerhenderson/architecture_decision_record
- Formless - No conventions for file format and structure

9.2.3 Decision Outcome

Chosen option: “MADR 2.1.0”, because

- Implicit assumptions should be made explicit. Design documentation is important to enable people understanding the decisions later on. See also [A rational design process: How and why to fake it](#).
- The MADR format is lean and fits our development style.
- The MADR structure is comprehensible and facilitates usage & maintenance.
- The MADR project is vivid.
- Version 2.1.0 is the latest one available when starting to document ADRs.

9.3 Java Framework

9.3.1 Context and Problem Statement

We want to use a well established Java framework in order to bootstrap and boost our development process.

9.3.2 Considered Options

- <https://spring.io/projects/spring-boot>
- <http://micronaut.io/index.html>
- <https://www.playframework.com>
- <https://akka.io>
- <https://www.dropwizard.io/1.3.5/docs>
- <https://projects.eclipse.org/proposals/eclipse-microprofile> with <https://openliberty.io>

9.3.3 Decision Outcome

We chose option 1, Spring Boot, since existing knowledge and experience with this framework is available. To reduce training costs (time) we chose this option.

9.4 Performance considerations

9.4.1 Context and Problem Statement

Should we consider and address performance requirements?

9.4.2 Decision Outcome

For the initial start of the project we do not consider performance as a must have.

9.5 Security considerations

9.5.1 Context and Problem Statement

Should we consider and address security requirements?

9.5.2 Decision Outcome

For the initial start of the project we do not consider security as a must have.

9.6 Monolith as Architectural Style

9.6.1 Context and Problem Statement

Which architectural style should we use to build MICO?

9.6.2 Considered Options

- Monolith: <https://martinfowler.com/bliki/MonolithFirst.html>
- Microservice Architecture: <https://martinfowler.com/articles/microservices.html>

9.6.3 Decision Outcome

For the initial project we apply the “Monolith first” approach in order to avoid the additional complexity of a microservice architecture.

9.7 Repository Structure Frontend/Backend

9.7.1 Context and Problem Statement

Should we use different repositories for Frontend and Backend or use a single repository?

9.7.2 Considered Options

- Mono Repo
- Different repositories

9.7.3 Decision Outcome

We chose option 1, Mono Repo, because as a starting point it reduces unnecessary complexity when working with frontend and backend logic. Furthermore, we can have a single CI build in order to verify the whole project source code.

9.8 Web-UI Framework

9.8.1 Context and Problem Statement

We want to use a well established UI framework in order to bootstrap and boost our development process for state-of-the-art web applications.

9.8.2 Considered Options

- Angular
- React

9.8.3 Decision Outcome

We chosen option 1, Angular, since existing knowledge and experience with this framework is available.

9.9 Database Technology (Relational/Graph)

Technical Story: [Evaluate database technologies #46](#)

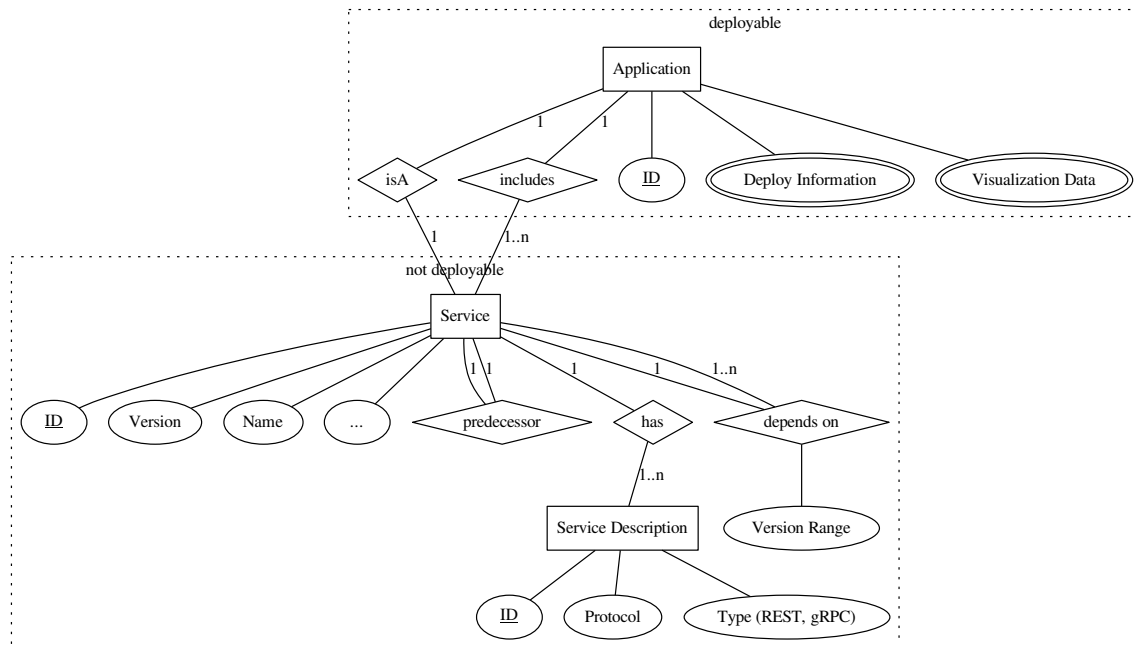
9.9.1 Context and Problem Statement

We want to use a state of the art database that can store services/applications according to our data Model and is easy to integrate with [Spring Boot](#).

9.9.2 Decision Drivers

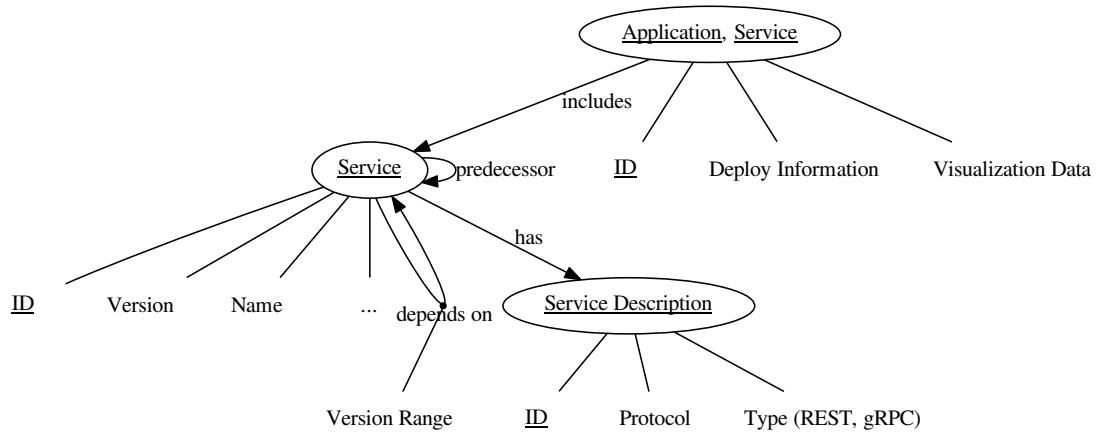
- Integration with [Spring Boot](#)
- Different versions of a service will be stored as individual services
- Applications will be seen as concrete instances of services

Data Model Relational



Important: The Service Description entity in the model describes a *Service Interface*.

Data Model Graph



Important: The `Service Description` entity in the model describes a *Service Interface*.

9.9.3 Considered Options

- Relational database:
 - MySQL
 - MariaDB
 - PostgreSQL
 - SQLite
- Graph database:
 - Neo4J

9.9.4 Decision Outcome

Chosen option: The graph database Neo4J, because it fits our data model best.

9.9.5 Pros and Cons of the Options

Relational database

- Good, because team already has sql knowledge
- Good, because Spring Boot integrations and ORM for java available
- Good, because sql scheme
- Bad, because sql scheme needs to be defined

- Bad, because data migrations to new database format need to update schema and data

Graph database

- Good, because applications and services naturally form a graph
- Good, because Spring Boot integrations and OGM for java available
- Good, because node/edge properties are dynamic
- Bad, because new query language
- Bad, because no rigid schema like in sql databases and therefore more necessary checks (e.g. type, exists) in code

9.10 Browser Compatibility

9.10.1 Context and Problem Statement

Which browser should we actively support with the MICO web interface?

9.10.2 Considered Options

- Chrome
- Firefox
- Edge
- Internet Explorer

9.10.3 Decision Outcome

We will only actively support Chrome (above version 70). We may test the web interface occasionally with Firefox (above version 63) to fix issues rendering the web interface unusable with firefox.

9.11 Features first

9.11.1 Context and Problem Statement

How should we prioritise certain non-/extra- functional requirements?

9.11.2 Considered Options

Requirements:

- Translatable text in user Interface
- Minimal security (username+password login for user interface/tokens for api)
- Responsive Web design

9.11.3 Decision Outcome

For the initial project we won't implement the requirements from above to increase our efficiency in implementing features.

See also:

- *Performance considerations*
- *Security considerations*

9.12 REST API Design

9.12.1 Context and Problem Statement

RESTful APIs can be designed differently, e.g., there are discussions about how to name resources correctly or what kind of HTTP methods should be used to trigger actions.

9.12.2 Decision Outcome

We follow the guidelines proposed by the [Learn REST: A RESTful Tutorial](#).

9.13 JSON+hal as serialization Format for the API

9.13.1 Context and Problem Statement

The backend needs to serialize the REST resources before sending them via http to the client.

9.13.2 Decision Drivers

- the format should be REST/HATEOAS compatible (e.g. allow hyperlinks)
- the format should be easy to parse/serialize for the backend and the client

9.13.3 Considered Options

- xml
- pure json
- json+hal

9.13.4 Decision Outcome

We will use json+hal (or the spring implementation for json with hyperlinks) without the `__embedded` attribute (because of its complexity to implement correctly).

See also:

- *REST API Design*

- Richardson Maturity Model
- HATEOAS is for Humans

9.14 Re-use capabilities of Winery Topology Modeler

Technical Story: Evaluate re-use capabilities of Winery Topology Modeler #32

9.14.1 Context and Problem Statement

We want to use a component to graphically model nodes and edges in a web application.

9.14.2 Decision Drivers

- usable with Angular
- number of dependencies
- features
 - create Nodes
 - create Connections
- usability

9.14.3 Considered Options

- Winery Topology Modeler
- ngx-graph
- d3.js, from React Version implemented in Angular

9.14.4 Decision Outcome

Chosen option: Chosen a Combination of ngx-graph and plain d3.js, because ngx-graph might miss some usefull features but has good usability which is worth to have a look at

Positive Consequences

- easy to use graph modeler
- only a few dependencies
- full control over features and behavior

Negative consequences

- needs to be implemented

9.14.5 Pros and Cons of the Options

Winery Topology Modeler

- Bad, since package doesn't work when importing into plain Angular App, created issue report on their repository

ngx-graph

- Good, because possibility to create templates for nodes, links
- Good, because runs as angular component
- Bad, because might not provide all the needed features

d3.js

- Good, because full control over everything
- Good, because can implement the necessary behavior
- Bad, because implementation has to be done

9.15 Source-to-Image Workflow

Technical Story: [Evaluate Knative build](#)

9.15.1 Context and Problem Statement

We want to have a Source-to-Image workflow to import services based on a GitHub repository. It should run inside our Kubernetes cluster, however currently Kubernetes doesn't have a resource build-in that is able to build container images. Therefore another technology is required.

9.15.2 Decision Drivers

- MUST run on our Kubernetes cluster
- MUST run completely in userspace (no root access required)
- MUST be sufficient to provide a single URL to a GitHub repository (with included Dockerfile)
- SHOULD be independent of any cloud service provider

9.15.3 Considered Options

- Gitkube
- Google Cloud Build
- Azure Container Registry Tasks (ACR Tasks)
- OpenShift Source-to-Image (S2I)
- Knative Build

9.15.4 Decision Outcome

Chosen option: *Knative Build*, because it meets all of our criterion decision drivers. It allows us to implement a Source-to-Image workflow on our Kubernetes cluster independently to any cloud service provider.

Positive Consequences

- By using *Knative Build* we have the choice to use different kinds of Builders, follow-up decision is required: [Building OCI images](#)

Negative consequences

- *Nothing known*

9.15.5 Pros and Cons of the Options

Gitkube

[GitHub: Gitkube](#)

- Good, because it is easy to use (only git and kubectl are required)
- Good, because it is designed to build and deploy images to Kubernetes
- Bad, because it uses Docker-in-Docker approach (see [Building OCI Images](#))
- Bad, because it is build for a different purpose: Helping developers to trigger the build and deployment of an image by using `git push`

Google Cloud Build

[Cloud Build documentation](#)

- Good, because it is easy to use, nothing has to be installed (only a call to the `Cloud Build API` is required)
- Good, because it is a managed service therefore it doesn't consume our own resources
- Good, because it allows us to use different Builder technologies (see [Building OCI Images](#))
- Bad, because we depend on the continuity of a third-party service (Google Cloud Build)
- Bad, because it runs only on Google Cloud, that forces vendor lock-in
- Bad, because it leads to additional costs (first 120 builds-minutes per day are free, see [Pricing](#))

Azure Container Registry Tasks (ACR Tasks)

[Tutorial: Automate container image builds with Azure Container Registry Tasks](#)

- Good, because it is a managed service therefore it doesn't consume our own resources
- Good, because we already run on the Azure Cloud
- Bad, because it uses the Docker daemon internally, no other Builders can be used (see [Building OCI Images](#))
- Bad, because there is no public API available, only usable with the Azure CLI
- Bad, because we depend on the continuity of a third-party service (ACR Tasks)

- Bad, because it runs only on Azure, that forces vendor lock-in
- Bad, because it leads to additional costs (100 minutes are included for free per month, see [Pricing calculator](#))

OpenShift Source-to-Image (S2I)

GitHub: [Source-To-Image \(S2I\)](#)

- Good, because it is the way to go for creating a Source-to-Image workflow on a OpenShift cluster
- Bad, because it is designed for the OpenShift Container Platform (additional effort is required to use it on plain Kubernetes)
- Bad, because it uses the Docker daemon internally (in future *Buildah*), no other Builders can be used (see [Building OCI Images](#))

Knative Build

GitHub: [Knative Build](#)

- Good, because it has the backing of industry giants (Google, Red Hat, IBM, SAP)
- Good, because it is designed for Kubernetes
- Good, because it provides a standard, portable, reusable, and performance optimized method for defining and running on-cluster container image builds
- Good, because it allows us to use different Builder technologies (see [Building OCI Images](#))
- Good, because *Knative Build* consumes little resources (2 pods a ~11 MB).
- Bad, because it is still work-in-progress

9.16 Evaluate Spring Data REST

Technical Story: [Evaluate Spring data rest](#)

9.16.1 Context and Problem Statement

We want to use CRUD operations to manipulate our domain model. Furthermore, we don't want to write REST controllers if it is not necessary.

9.16.2 Decision Drivers

- Must allow CRUD operations
- It must be possible to extend the API and to add resources or sub resources which are not managed by Neo4j
- It must be compatible with Springfox
- The API must be RESTful

9.16.3 Considered Options

- Spring Data REST
- Manually writing Spring REST controllers (based on Spring HATEOAS)

9.16.4 Decision Outcome

Chosen option: Spring Data REST, because it automatically exposes our domain model and so we don't have to write the required rest controllers our self. In addition to that paging, sorting and the HAL media type are supported out-of-the-box. Extending or overwriting Spring Data REST Response Handler is also possible and allows us to add custom logic if needed [1].

9.16.5 Pros and Cons of the Options

Spring Data Rest

- Good, because our model is exposed automatically
- Good, because it supports paging, sorting and HAL
- Good, because changes in the model are automatically reflected in the API
- Bad, because we have to use a nonstable version of Springfox (3.0.0-SNAPSHOT) [2]
- Bad, because we need to tweak the generated API to map some features of our domain model nicely (e.g. service1->dependsOn->service2->...)

Manuel approach

- Good, because we have more control over the resources which are offered by our API
- Good, because it is easier to hide internal changes to offer a more stable API
- Bad, because it requires more manual work

9.17 Building OCI Images

Part of the Technical Story: [Evaluate Knative build](#) Related to the [ADR-0013 Source-to-Image Workflow](#).

9.17.1 Context and Problem Statement

We want to build OCI images based on Dockerfiles inside our Kubernetes cluster.

9.17.2 Decision Drivers

- MUST run completely in userspace (no privileged rights required)
- MUST be runnable on Kubernetes
- MUST be controlled in an automatic manner (CI/CD)
- SHOULD be compatible with Knative Build

9.17.3 Considered Options

- Docker out of Docker
- Docker in Docker
- BuildKit
- img
- Jib
- Buildah
- Kaniko

9.17.4 Decision Outcome

Chosen option: Kaniko, because it is designed for the use case we need. It works on Kubernetes and build OCI images without any daemon and without privileged rights.

9.17.5 Pros and Cons of the Options

Docker out of Docker

- Good, because Docker daemon is already running in the Kubernetes cluster
- Bad, because it would interfere the Kuberentes scheduling
- Bad, because it requires privileged mode in order to function, which is a significant security concern
- Bad, because it is not compatible with *Knative Build*

Docker in Docker

- Good, because it is a well known approach
- Bad, because it requires privileged mode in order to function, which is a significant security concern (still better than *Docker out of Docker*)
- Bad, because it generally incurs a performance penalty and can be quite slow
- Bad, because it is not compatible with *Knative Build*

BuildKit

GitHub: BuildKit

- Good, because it is created by Docker (Moby) as a “next generation `docker build`”
- Good, because it is already successfully used by the OpenFaaS Cloud
- Good, because it can run on Kubernetes without privileged rights
- Good, because an official *Knative Build Template* exists
- Bad, because it needs a deamon

img

GitHub: [img](#)

- Good, same as *BuildKit* but daemonless
- Bad, because it needs a special Linux Capability “[RawProc](#) to create nested containers
- Bad, because currently no official *Knative Build Template* exists (nevertheless it could be created ourself)

Jib

GitHub: [Jib](#)

- Good, because it is fast and daemonless
- Good, because an official *Knative Build Template* exists
- Bad, because it is designed for Java applications only

Buildah

GitHub: [Buildah](#)

- Good, because it is a popular tool for creating OCI container images and maintained by Red Hat (also included by default in RHEL)
- Good, because it is daemonless
- Good, because an official *Knative Build Template* exists
- Bad, because it requires privileged rights (same as Docker daemon), rootless mode is planned

Kaniko

GitHub: [kaniko](#)

- Good, because it is a popular open source project created by Google
- Good, because it is well documented and easy to use (readymade executor image `gcr.io/kaniko-project/executor` exists)
- Good, because it is designed to work on Kubernetes
- Good, because it is daemonless
- Good, because it does not need privileged rights (however it requires to run as root inside the build container)
- Good, because an official *Knative Build Template* exists
- Bad, because there are security concerns about malicious Dockerfiles due to the lack of isolation [Issue #106](#)

9.18 Documentation Structure

9.18.1 Context and Problem Statement

We want to have a well defined structure for what to document where.

9.18.2 Considered Options

- Everything in docs repository
- Everything in mico repository
- Leave it as is (mostly in docs repository & code documentation in mico repository)

9.18.3 Decision Outcome

Chosen option: Leave it as is.

Clarification for what to document where:

Docs Repository

Rationale: If I want to use mico this documentation should contain all I need to know.

Examples:

- Design Decisions
- Architecture
- Component descriptions (mico-core, mico-admin, mico-grapheditor)
- API Interfaces (REST Api, etc.)
- How To's (How To install mico, How To use mico, etc.)

Mico Repository

Rationale: If I want to change something in the code this documentation should contain additional information about code specifics.

- Javadoc, Typedoc, etc.
- Descriptions for important classes and how to use them
- How to write a new REST Endpoint, Unit Test, Plugin, etc.
- Links to dependencies (with documentation), relevant documentation in docs repo

If possible use links to the relevant documentation instead of describing the whole thing.

9.18.4 Everything in docs repository

Docker out of Docker

- Good, because of clear separation of documentation and code
- Bad, because sphinx can use language native documentation (like javadoc) only if code is in the same repository

Everything in mico repository

- Good, because one repository for everything
- Bad, because the workflow for documentation (without pull requests) and the workflow for code changes (with pull requests) oppose each other

Leave it as is (mostly in docs repository & code documentation in mico repository)

- Good, because it enables fast documentation workflow without pull requests
- Good, because separate code documentation still supports javadoc, etc.
- Good, because built documentation still feels like one if interlinked properly
- Bad, because it needs explanation for what goes where

9.19 Istio vs. Envoy

Part of the Technical Story: [Evaluate Istio vs. Envoy vs. xxx](#)

9.19.1 Context and Problem Statement

We want to have telemetry (performance, monitoring) and log collections for each MICO service that should be added automatically during deployment.

9.19.2 Decision Drivers

- MUST be compatible with Kubernetes
- MUST at least provide all required features (telemetry, logs)
- MUST be independent of the technology used in the respective service
- SHOULD consume less resources

9.19.3 Considered Options

- Istio
- Envoy

9.19.4 Decision Outcome

Chosen option: Istio, because it is easier to configure than only using Envoy. By using Istio the Envoy proxies can be used as the data plane and Istio as the control plane to manage the whole service mesh. Moreover Istio is the better choice for the future, because it has all features that could be potentially helpful for the management of a service mesh.

Positive Consequences

- Istio enables more than just having telemetry and log collections. In addition it provides many more features that could be useful for MICO in future (follow-up decisions required).

Examples:

- Automatic Load Balancing
- Routing rules, retries, failovers, and fault injection
- Pluggable policy layer with access controls, rate limits and quotas
- Secure service-to-service communication in the cluster

Negative consequences

- Istio consumes many resources
- Istio brings more complexity into the MICO system

9.19.5 Pros and Cons of the Options

Istio

- Good, because it provides the control plane to configure the Envoy proxies (sidecars)
 - Istio Proxy (extended version of Envoy) is easier to configure than plain Envoy
 - It is integrated into Kubernetes with Custom Resource Definitions, therefore Istio is configurable with the same YAML DSL than Kubernetes
- Good, because it provides additional functionality that could be relevant in future (e.g. access control, secure communication, routing rules)
- Bad, because Istio has a steep learning curve to get familiar with all its features
- Bad, because it is still in development (though the current version is 1.0.5)

Envoy

- Good, because it is a high-performance proxy with a lightweight footprint (developed in C++)
- Good, because it a graduated project of the Cloud Native Computing Foundation (next to Kubernetes and Prometheus)
- Bad, because the YAML DSL for configuration of Envoy differs from that of Kubernetes, which we are accustomed to use
- Bad, because in a service mesh it's complex and error prone to configure

9.20 Requirements regarding the Composition of Applications

9.20.1 Context and Problem Statement

We want to have clear and simple requirements when it comes to the way applications can be created in the user interface.

9.20.2 Decision Drivers

- MUST be compatible with Lombok

9.20.3 Considered Options

- Applications cannot be composed out of other applications. Applications can only be created using single services.
- Applications can be part of other (new) applications. Therefore inheritance between MicoApplication and MicoService is necessary. This would mean difficulties with Lombok.

9.20.4 Decision Outcome

Chosen option: the first option, since we want a simple solution, in order to have the system running as soon as possible.

Positive Consequences

- Lombok can be used.
- Better code quality.

Negative consequences

- Applications cannot be created using other existing applications.

9.21 Architecture for background jobs

Part of Technical Story: [<https://github.com/UST-MICO/mico/issues/95>]

9.21.1 Context and Problem Statement

Several operations (crawl repo, deploy services, ...) are required to run in the background. Therefore, a background job architecture is required being able to launch arbitrary jobs in the background.

9.21.2 Decision Drivers

- MUST be implemented in Java
- MUST offer a callback functionality

9.21.3 Considered Options

- Java ExecutorService
- Java CompletableFuture
- Spring Scheduling

9.21.4 Decision Outcome

Chosen option: Java CompletableFuture, because of the possibility to chain multiple actions and the option to attach callbacks on completion of a background task.

Positive Consequences

- Possibility to chain together multiple actions after a successful execution of a background task.
- Use of FixedThreadPool, i.e., a pool of threads with a predefined number of threads which are reused. If all threads are active at a time and additional tasks are submitted, they will wait in a queue.

Negative Consequences

- With the use of a FixedThreadPool, a background task may be delayed if all threads are active at the time of submission. This may produce time of waiting in the frontend of MICO.

9.21.5 Pros and Cons of the Options

Java ExecutorService

- Good, because it offers a simple interface.
- Good, because it enables basic functionality for async background task execution and scheduling.

Java CompletableFuture

- Good, because of possibility to chain multiple actions.
- Good, because built-in Java feature.
- Good, because callbacks can be attached to be executed on completion.

Spring Scheduling

- Good, because of easy implementation as a result of existing spring injections implemented in the application.
- Good, because well suited for monitoring and management (providing several useful attributes).
- Bad, because of no known possibility to receive a feedback of an async executed action.

9.21.6 Links

9.22 Configurable Service Dependencies

9.22.1 Context and Problem Statement

Services currently have a static network of dependencies. This makes it impossible to have a service that depends on a SQL based database use a different database that is compatible with the service.

9.22.2 Decision Drivers

- Should fit in the current domain model
- Easy to add afterwards
- Should support existing services with their static dependencies

9.22.3 Considered Options

Let service specify options for dependencies

Let the user specify if the dependency is optional and/or what other services may be used to fulfill the dependency.

Pro: High control over allowed alternatives.

Contra: Can't capture all alternatives.

Specify if services are a compatible alternative to a different service

Let the user specify if a service can be used as alternative to an existing service.

Maybe add meta-services that are similar to meta-packages in linux package managers.

Pro: Works for all existing services.

Contra: There may still be incompatibilities.

Add service groups for services that are interchangeable

Same as above but with different implementation.

Let the user define service groups to group services that can be used interchangeably.

Contra: Needs dependency edges to group objects that are not deployable (=> more implementation work)

Hardcode some service groups

Hardcode some service groups (e.g. Database services). The service groups can have special properties customized for the included services. The services can still be added to the hardcoded groups by a user.

9.22.4 Decision Outcome

To be decided.

This adr purely documents possible decisions for this problem.

9.23 Kafka as Messaging Middleware

9.23.1 Context and Problem Statement

To compose service with the mico application we need a messaging middleware that is highly scalable and supports the [Enterprise Integration Patterns](#).

9.23.2 Decision Drivers

- Should be a well known and proven solution
- Support a pipes and filters style architecture

9.23.3 Considered Options

Apache Kafka

Pro: Proven system with high scalability. Well suited for pipes and filters architectures.

Contra: Topics have different semantics than traditional message queues. Some patterns don't work well/at all with the topic semantic.

Some other messaging middleware

This option was not explored further.

9.23.4 Decision Outcome

We want to use Kafka as our messaging middleware.

Important things to know about Kafka

Kafka topics have a different semantic than traditional message queues. Kafka also does NOT log whether a message was consumed. It only stores the last committed offset for a consumer. Because of that and the different topic semantic some patterns need to be adapted to work with Kafka.

For more detailed information about the inner workings of Kafka consult the following links:

9.24 Function to component mapping

Technical Story: [Issue #713](#)

9.24.1 Context and Problem Statement

To implement the EAI patterns we use a combination of a generic component which handles the communication with Kafka and a FaaS solution. The business logic of the EAI patterns (message splitting/aggregation or transformation) is provided via functions which are hosted on the FaaS solution. The generic component communicates with Kafka and de/serializes the messages. We need a means to wire the instances of the generic component with the functions. E.g. A user wants to insert a message splitter between two message-based components. To realize this an instance of the generic component in combination with a splitting FaaS function will be used. The generic component needs the address of the FaaS gateway and the function name (e.g. <http://address:8080/function/msg-payload-splitter>) to call the function. To provide the necessary information to instances of the generic component we considered the following techniques.

9.24.2 Decision Drivers

- MUST be supported by the language/technology which is used to implement the generic component
- MUST be easy to integrate into MICO
- SHOULD be a well known and proven solution

9.24.3 Considered Options

- Environment variables
- Function registry
- Control Topic
- Function composition
- Function proxy
- Configuration file

9.24.4 Decision Outcome

Chosen option: “Environment variables”, because MICO already supports this and it is easy to implement in the generic component.

9.24.5 Pros and Cons of the Options

Environment variables

This option uses environment variables to store the necessary information. These variables are controlled by MICO.

- Good, because MICO already supports setting environment variables
- Good, because most languages support reading environment variables
- Good, because we need environment variables anyway because we need to provide the Kafka configuration
- Good, because it is a well known and proven solution
- Good, because it follows the (twelve-factor app methodology)[<https://12factor.net/de/config>]
- Bad, because the configuration can not be changed on the fly (do we want this?)

Function registry

We implement a separate component which stores the mapping of the functions to the instances of the generic component. We could evaluate if (Apache ZooKeeper)[<https://zookeeper.apache.org>] or (etcd)[<https://github.com/etcd-io/etcd>] fits this purpose.

- Good, because the mapping can change in the fly
- Good, because there is support for this approach in most languages
- Bad, because more development/evaluation overhead

Control Topic

We already use Kafka so we could follow the [dynamic router pattern](#) and use a control channel(topic) to configure the instances. This topic could use the [compaction feature](#) to always retain the last known configuration for each message key.

- Good, because we already decided to use Kafka
- Good, because this option provides a log of configuration changes
- Bad, because the configuration for Kafka could not be provided via this method
- Bad, because more development/evaluation overhead

Function composition

We don't implement the generic component as a separate component but as a function which is hosted on the FaaS solution. There is a [Kafka connector](#) for OpenFaaS. Then compose the functions with a composer like [this](#) or follow the workflow techniques described in the OpenFaaS [documentation](#).

- Good, because the basic components for this approach already exist
- Good, because no separate generic component is necessary
- Bad, because the composer does not seem to be very mature and proven

Function proxy (API Gateway)

Each instance of the generic component calls the API gateway which handles the routing/mapping to the functions.

- Good, because each instance of the generic component calls the same gateway and therefore needs the same configuration
- Bad, because we don't have this proxy and therefore this adds development overhead

Configuration file

Generate a configuration file per instance of the generic component and deploy it with the instance. Same pros and cons like environment variables but more complexity for file generation/parsing and deployment.

9.25 FaaS

9.25.1 Context and Problem Statement

To execute a variety of different functions that are based on different messaging patterns we want to use the Function as a Service (FaaS) approach.

9.25.2 Decision Drivers

- MUST support Kubernetes
- MUST support Kafka to trigger events
- SHOULD be a well known and proven solution

- SHOULD support the most common programming languages (at least Java, Python, JavaScript)
- SHOULD support the Serverless framework

9.25.3 Considered Options

9.25.4 Decision Outcome

We want to use OpenFaaS.

9.25.5 Pros and Cons of the Options

OpenFaaS

- Good, because it supports Kubernetes
 - via [faas-netes](#) (installation via `Helm` or `kubectl`)
 - via [OpenFaaS Operator](#)
- Good, because it supports the Serverless framework
- Good, because it is an easy to use Serverless framework
- Good, because it has a simple architecture and can be well understood
- Good, because it supports Kafka via the [Kafka-connector](#)

OpenWhisk

- Good, because it supports Kubernetes (installation via `Helm`)
- Good, because it supports the Serverless framework
- Good, because it is a mature Serverless framework supported by the Apache foundation and backed by IBM
- Bad, because it is more complex than OpenFaaS and it leverages many components (CouchDB, Kafka, Nginx, Redis and Zookeeper)

Kubeless

- Good, because it integrates natively into Kubernetes as a CRD
- Good, because it supports the Serverless framework
- Good, because it doesn't require extra tooling (`kubectl` is enough)
- Good, because it uses Kafka messaging system as event triggers
- Bad, because it isn't as mature as OpenFaaS or OpenWhisk, documentation isn't as good as the others

Knative Serving

- Good, because it is baked by Google
- Good, because it supports Kubernetes
- Good, because Knative Eventing supports Kafka events
- Bad, because it requires Istio
- Bad, because it is more complex than e.g. OpenFaaS

9.26 Language for a generic composition pattern implementation

9.26.1 Context and Problem Statement

We need to decide which language to use for implementing a MICO composition service. The framework will be [Apache Kafka](#) See [ADR0021](#)

9.26.2 Decision Drivers

- Should be known by everyone (on the development team)
- Must support Apache Kafka

9.26.3 Considered Options

Java

Pro:

- Easy setup with [Spring boot](#)
- Java is the only language in which every team member is proficient in

Python

Python was only used in the prototyping stage and was not explored further

9.26.4 Decision Outcome

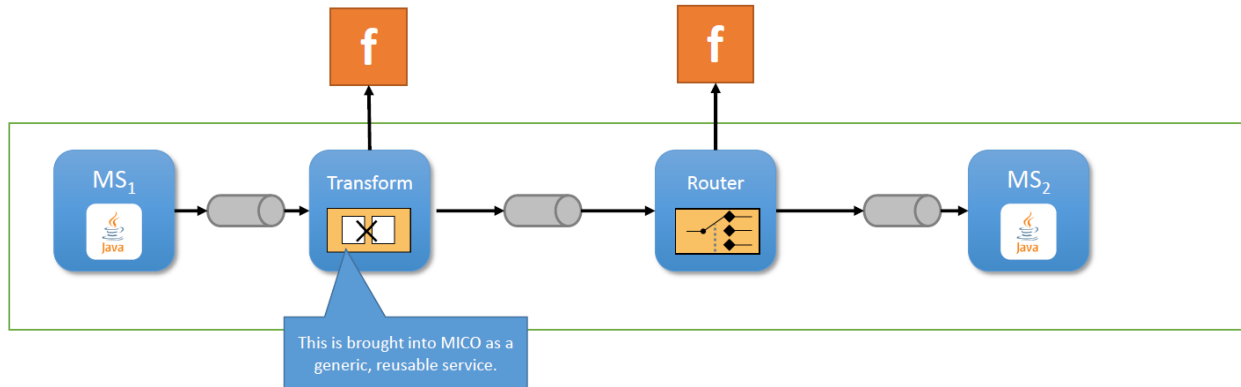
We want to use Java. Since existing knowledge and experience is given for everyone.

9.27 [WIP] Kafka-faas-connector requirements (aka generic component requirements)

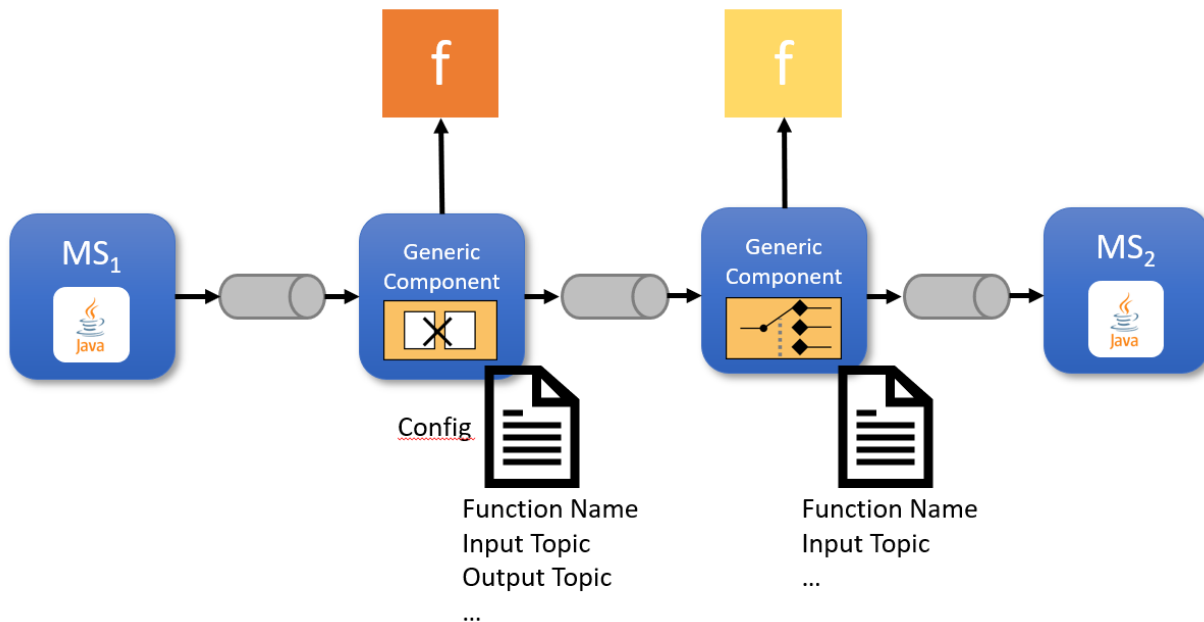
Technical Story: [<https://github.com/UST-MICO/mico/issues/724>]

9.27.1 Context and Problem Statement

We want to realize the following user story: As a user, I want to connect two components of a distributed app. Both components write/listen to different topics and use a different message format. To realize this, a message router and a message transformer (EAI Pattern) is needed as depicted in the following sketch.



The transformation and the routing logic is provided via functions which are hosted on a FaaS solution. These functions are user specific and could be provided via a function store. The transformer doesn't need to know how to transform/split/enrich/filter the messages. It simply reads the message from a specific topic, hands it over to the function and sends the (transformed) result of the function to an output topic. Likewise, the router does not need to know how to route the messages (content or context based, e.g. if `itemPrice > 50000` then route to high-value department). It reads the message from a topic, hands it over to a function and sends the message to a function determined topic. Both components have in common that they only read messages from a topic, hand the message to a function and act upon the result. We could draw the image above like this:



9.27.2 Considered Options

- A generic message processor component
- OpenFaaS Kafka connector

9.27.3 Decision Outcome

Chosen option: Kafka-faas-connector, because we don't want to mix messaging and business logic in the functions and need to have dynamic topics which are not supported by the OpenFaaS Kafka connector at the moment.

9.27.4 Detailed Option description

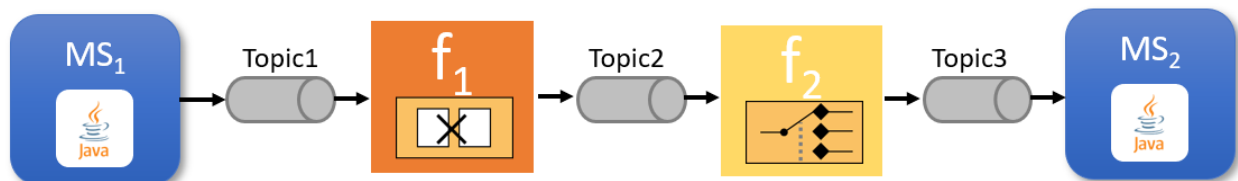
Kafka-faas-connector (aka generic message processor component)

As described above we could solve this with a generic message processor component which reads messages from an input topic, hands them over to a function and then acts upon the results. This could either be to post the transformed message to an output topic (message transformer) or to send the original message to a function determined output topic (message router). This generic message processor needs the following capabilities:

- Read messages from a Kafka topic
- Call a function hosted on a FaaS solution with a message payload
- Receive zero, one or multiple (splitter) messages from the function
- Call a function with a message payload and read the [routing slip attributes](#) of all returned messages
- Route each returned message according to its routing slip
- Send messages to a function or configuration determined topic
- If there was an error during processing, then send the message to an Invalid Message Topic
- If there was an error during routing (topic does not exist), then send the message to a Dead Letter Topic
- Attach a message history to the message. The history contains the name of the generic component and the name of the configured FaaS function.
- Use an HTTP interface for debugging and metrics (Spring actuator)
- TODO State handling for aggregator and other stateful message patterns

OpenFaaS Kafka connector

In contrast to the generic message processor component approach, we could use the [Kafka connector](#) to read messages from topics. We could transform the first image on the fly into something like this:



The functions are still user provided but we deploy or update the functions with annotations which contain the input topic (e.g. for f1 we would add `topic=Topic1`). The functions get the output topic via an environment variable or determine it them self (in case of a router).

Example function yaml

```

provider:
  name: openfaas
  gateway: http://127.0.0.1:8080
functions:

```

(continues on next page)

(continued from previous page)

```
msg-payload-transformer:
  lang: java8
  handler: ./msg-payload-transformer
  image: msg-payload-transformer:latest
  environment:
    output_topic: Topic2
  annotations:
    topic: Topic1
```

Pros and cons:

- Good, because we don't need the generic message processor
- Good, because we save a round trip from the processor to the function
- Bad, because we need to know all used topics beforehand ([Issue #2](#))
- Bad, because each function has to handle message sending
- Bad, because we mix business logic in functions with messaging logic

9.28 Implementation of complex EAI-Patterns with FaaS

9.28.1 Context and Problem Statement

Some [Enterprise Integration Patterns](#) have a complex structure where parts of the behaviour can be implemented generically while some parts need to be modifiable by the end user (in our case the system admin using MICO). We have already [decided to use a FaaS platform](#) to provide this modifiability in form of code as configuration. While this works well for most patterns, for some of the more complex patterns it is not easy to allow modifiability via FaaS. This is especially the case if the user want to write as little code as possible meaning the [generic part](#) of the component has to be implemented by the MICO team.

9.28.2 Decision Drivers

- Modifiability of the patterns must be provided via a FaaS function
- The function should only have to contain as little code as possible
- Generic code for the pattern should be provided by the MICO platform either as a library to import into the FaaS function or in the component that calls said function

9.28.3 Challenges

Where / how to implement generic logic

- Implement all logic in the FaaS function
- Implement custom logic + some of the generic logic in the FaaS function
- Implement only custom logic in the FaaS function

State of configuration channel

How can the FaaS function get the current state of the configuration (e.g. dynamic router).

- Let the FaaS function subscribe to Kafka
- Have a separate DB
- Send the current configuration together with the message

Sending to multiple destinations / unknown destinations

A router does not have a specific endpoint to send all messages to and may even send messages to many endpoints.

- Implement generic routing with `recipientList` / `routingSlip` only and let FaaS function write the route into the message header
- Let the function send the messages directly to Kafka

Stateful components

Some patterns are inherently stateful.

- Store state in a DB
- Store state in generic component and send to FaaS function with message

Performing intermediate request steps

A content enricher needs to perform requests to get the content to inject into the message.

- Offload implementation completely to the user
- Split into two parts, one determining which requests are needed and one injecting the requested content into the message.

9.28.4 Affected patterns

| Pattern | dynamic config | destinations | stateful |
|----------------------|----------------|--------------|----------|
| Router | no | yes | maybe |
| Dynamic Router | yes | yes | maybe |
| Content based Router | maybe | yes | maybe |
| Aggregator | no | no | yes |
| Resequencer | no | no | yes |
| Process Manager | maybe | yes | maybe |
| Message Normalizer | maybe | no | no |

9.28.5 Decision Outcome

Where to implement logic: To be decided

State of configuration channels: To be decided

Stateful functions: To be decided

Routing: We will support custom routing decisions in the FaaS function by always interpreting a routing slip if it is present. The routing slip has to support multiple destinations for one routing step. This will also allow us to make more patterns possible (everything that is not stateful) with a single generic kafka to FaaS connector.

Intermediate requests: To be decided

9.28.6 Evaluation of Options

Where to implement generic logic

Implement generic logic in FaaS-Function

- Good because we can use Kafka as trigger for the function
- Good because it allows easy extension and modification of the generic logic where needed
- Bad because we need to implement the generic logic in all supported languages as a library/framework
- Bad because the alternative would mean complex FaaS networks/dependencies if generic logic is capsuled in FaaS functions

Implementing generic logic in Kafka to FaaS Adapter

- Good because we only need one implementation in one language
- Bad because generic logic needs to support all patterns => possibly more complex implementation

Stateful patterns

Store state in compacted Kafka topic

- Good because it uses no extra component
- Bad because only useful for storing configuration channel state/state that does not change too often
- Bad because it needs many assumptions
- Bad because whole channel needs to be read to load state

Store state in DB

- Good because DBs are really good for storing and querying state
- Bad because management of DBs (isolation of DBs for different FaaS functions) is difficult

Send state with the message to the FaaS function

- Good because FaaS function can be trivially stateless
- Bad because state could be large
- Bad because generic component needs to know what state to send (up priori assumption)

Sending to multiple or unknown destinations

Allow FaaS function to send messages directly to kafka

- Good because it allows complex routing patterns
- Bad because FaaS function needs to understand and react to various header fields in the cloud event (routingSlip, route history)

Implement generic routing via header fields (routingSlip/recipientList)

- Good because we can ensure semantics of some header fields in our implementation
- Bad because header fields need to be sufficiently complex to support complex routing decisions

Intermediate requests

Offload complete implementation to user

- Good because all requests can be controlled by user code
- Bad because user may need to (re-)implement generic code
- Bad because FaaS function needs to know about kafka topics

Split into request part and merge part

- Good because split implementation is easier to understand
- Good because split implementation allows for generic request handling implemented by MICO
- Bad because split pattern implementation is more complex

9.29 Simple Composition Components

Technical Story:

- Depends on:
- Related issues:

9.29.1 Context and Problem Statement

A Simple Composition Component is a [Generic Component](#), that follows the concept of a certain [Enterprise Integration Pattern](#) by invoking a user-defined (or predefined) openFaaS function and acting upon its results. In contrast to the [Complex Integration Components](#) there are no further efforts (such as maintaining a state) needed for implementing the corresponding Simple Composition Component.

9.29.2 Decision Drivers

We want to have a list of all Enterprise Integration Patterns, that can be implemented as Simple Composition Components with a reasonable amount of effort.

9.29.3 Considered Options

Theoretically, all Enterprise Integration Pattern could be implemented as Simple Composition Components

9.29.4 Decision Outcome

Which of those do we want to implement??

Routing Patterns

The following tables describes for all Routing, Transformation and Management patterns, if they can be implemented as Simple Composition Components (As defined above).

The following assumptions are made:

- We have two Generic Components:
 - A Generic Transformation Component: Provide each message to a user-defined function and forward the result to a predefined target topic
 - A Generic Routing Component: Provides each message to a user-defined function and forward the result to the topic, that is returned by the user-defined function.
- Both components read from a single topic only
- There are no external resources available (e.g. for storing a state)

Transformation Patterns

System Management

For new ADRs, please use *template.md* as basis.

See also:

More information on MADR is available at <https://adr.github.io/madr/>.

General information about architectural decision records is available at <https://adr.github.io/>.

9.30 Open Decisions

This section contains open decisions that may not have an ADR yet but are relevant for the project in the (near) future. Each decision should be a list item with a short description.

- Application Graph Editor: Winery Topologymodeler OR custom d3 editor

MICO Application An application is a set of services.

MICO Service MICO Service is the resource that is managed by the MICO system, is part of the MICO domain model and relates to the term “micro-service”. Such a MICO Service originates either from a GitHub repository or from DockerHub.

Kubernetes Service Kubernetes Service “is an abstraction which defines a logical set of Pods and a policy by which to access them - sometimes called a micro-service”. Pods themselves “serve as unit of deployment, horizontal scaling, and replication”. (quoted by Kubernetes documentation) The relationship between a MICO Service and a Kubernetes Service is quite confusing because of following association:

- MICO Service → Kubernetes Deployment
- MICO Service Interface → Kubernetes Service (MICO Service Interface is a different resource within the mico system, but is part of a MICO Service, actually it is a 1:n relationship)

Service Layer The Service Layer includes *Infrastructure Services* that are used by the domain entities. For example for the use-case of importing MICO services based on GitHub repositories the service *GitHubCrawler* exists. More examples are the *ImageBuilder* for creating images for MICO services, the *MicoStatusService* for retrieving data from Prometheus and *MicoKubernetesClient* as the service to operate with Kubernetes.

Broker Broker is part of the domain layer that extends the domain model with more business logic. A Broker adds operations to the model in ubiquitous language (not CRUD, that would be a Repository), but is completely stateless.

Service An application service retrieve the information required by an entity, effectively setting up the execution environment, and provide it to the entity. For example for the use-case of importing MicoServices based on GitHub repositories the Application Service *GitHubCrawler* exists.

External Service An external service is a service not managed by the mico system but by a third party, e.g. a weather api from weatherbit which could be used by a MICO Service. External services are currently not supported.

Service Interface A service interface is an API (REST over http, gRPC, etc.) the service provides for other services to use over the network. One MICO service can have multiple service interfaces, but at least one. A MICO service interface is represented through a Kubernetes Service.

Kafka ‘**Kafka**<https://kafka.apache.org>’_ is a MOM(Message oriented Middleware) that uses topics instead of traditional message queues.

Topic Messages that are send over Kafka are organized in kafka topics. A service has multiple topics (input, output, dead letter, etc) from where it can consume messages or push new messages.

OpenFaaS-function A OpenFaaS-function is used for manipulating messages based on different messaging patterns using the FaaS(Function as a Service) approach.

KafkaFaaSConnector The Kafka FaaS Connector is used by MICO as a generic message processor component to route messages from Kafka in the CloudEvents format to an OpenFaaS function.

HTTP Routing Table

/actuator

GET /actuator, 28
GET /actuator/configprops, 28
GET /actuator/health, 28
GET /actuator/health/{component}, 28
GET /actuator/health/{component}/{instanceId}, 28
GET /actuator/info, 29
GET /actuator/loggers, 29
GET /actuator/loggers/{name}, 29
GET /actuator/metrics, 29
GET /actuator/metrics/{requiredMetricName}, 30
POST /actuator/loggers/{name}, 29

/applications

GET /applications, 30
GET /applications/{str:shortName}/{str:version}, 66
GET /applications/{shortName}, 32
GET /applications/{shortName}/{version}, 34
GET /applications/{shortName}/{version}/deploymentInformation/{serviceShortName}, 37
GET /applications/{shortName}/{version}/services, 39
GET /applications/{shortName}/{version}/status, 41
POST /applications, 31
POST /applications/{str:shortName}/{str:version}/deploy, 66
POST /applications/{shortName}/{version}/deploy, 36
POST /applications/{shortName}/{version}/promote, 38
POST /applications/{shortName}/{version}/services/{serviceShortName}/{serviceVersion}, 41
PUT /applications/{shortName}/{version}, 35

PUT /applications/{shortName}/{version}/deploymentInformation, 37
DELETE /applications/{shortName}, 33
DELETE /applications/{shortName}/{version}, 36
DELETE /applications/{shortName}/{version}/services, 40

/cluster

GET /cluster/*, 66

/error

HEAD /error, 43
GET /error, 43
POST /error, 44
PUT /error, 44
DELETE /error, 44
PATCH /error, 45
OPTIONS /error, 45

/jobs

GET /jobs, 45
GET /jobs/{str:id}/, 65
GET /jobs/{id}, 46
GET /jobs/{shortName}/{version}/status, 47
DELETE /jobs/{id}, 46

/services

GET /services, 47
GET /services/, 63
GET /services/{str:shortName}/, 64
GET /services/{str:shortName}/{str:version}/, 64
GET /services/{str:shortName}/{str:version}/dependencies, 65
GET /services/{str:shortName}/{str:version}/dependencies, 65
GET /services/{str:shortName}/{str:version}/interface, 64

```
GET /services/(str:shortName)/(str:version)/dependees,
65 54
DELETE /services/{shortName}/{version}/dependees/,
65 55
GET /services/import/,65 DELETE /services/{shortName}/{version}/interfaces/,
GET /services/import/github,49 60
GET /services/{shortName},50
GET /services/{shortName}/{version},51
GET /services/{shortName}/{version}/dependees,
53
GET /services/{shortName}/{version}/dependencyGraph,
55
GET /services/{shortName}/{version}/dependers,
56
GET /services/{shortName}/{version}/interfaces,
57
GET /services/{shortName}/{version}/interfaces/{serviceInterfaceName},
59
GET /services/{shortName}/{version}/interfaces/{serviceInterfaceName}/publicIP,
61
GET /services/{shortName}/{version}/status,
62
POST /services,48
POST /services/,64
POST /services/(str:shortName)/(str:version)/dependees/,
65
POST /services/(str:shortName)/(str:version)/dependers/,
65
POST /services/(str:shortName)/(str:version)/interfaces/,
65
POST /services/import/(str:type)/,65
POST /services/import/github,49
POST /services/{shortName}/{version}/dependees/{dependeeShortName}/{dependeeVersion},
54
POST /services/{shortName}/{version}/interfaces,
58
POST /services/{shortName}/{version}/promote,
61
PUT /services/(str:shortName)/(str:version)/,
64
PUT /services/(str:shortName)/(str:version)/interfaces/(str:shortName),
65
PUT /services/{shortName}/{version},52
PUT /services/{shortName}/{version}/interfaces/{serviceInterfaceName},
59
DELETE /services/(str:shortName)/(str:version)/dependees/(str:shortName),
65
DELETE /services/(str:shortName)/(str:version)/dependers/(str:shortName),
65
DELETE /services/(str:shortName)/(str:version)/interfaces/(str:shortName),
65
DELETE /services/{shortName},51
DELETE /services/{shortName}/{version},
53
```

B

BACKEND_REST_API, [141–143](#)

Broker, [195](#)

E

environment variable

 BACKEND_REST_API, [141–143](#)

External Service, [195](#)

K

Kafka, [196](#)

KafkaFaaSConnector, [196](#)

Kubernetes Service, [195](#)

M

MICO Application, [195](#)

MICO Service, [195](#)

O

OpenFaaS-function, [196](#)

S

Service, [195](#)

Service Interface, [195](#)

Service Layer, [195](#)

T

Topic, [196](#)