Accelerate your journey to Kubernetes with the Konveyor Community

A community of people passionate about helping others modernize and migrate their applications to the hybrid cloud by **building tools and best practices on how to replatform and refactor applications to run on Kubernetes and cloud-native technologies**

KONVEYOR

A CNCF sandbox project

CLOUD NATIVE COMPUTING FOUNDATION

www.konveyor.io

Konveyor Community Projects

Hub

Inventory and Planning

Move2Kube

Replatform to K8s

Test Generator

Automate Test Generation

Containerization Advisor

Assess your Tech stack

DiVA

Data-intensive Validity Analyzer

Data Gravity Insights

Identify Transactional Boundaries

Pathfinder

Assess your application

Windup

Assess your source code

Pelorus

Measure software delivery performance

Replatforming to Kubernetes

with

Move2Kube

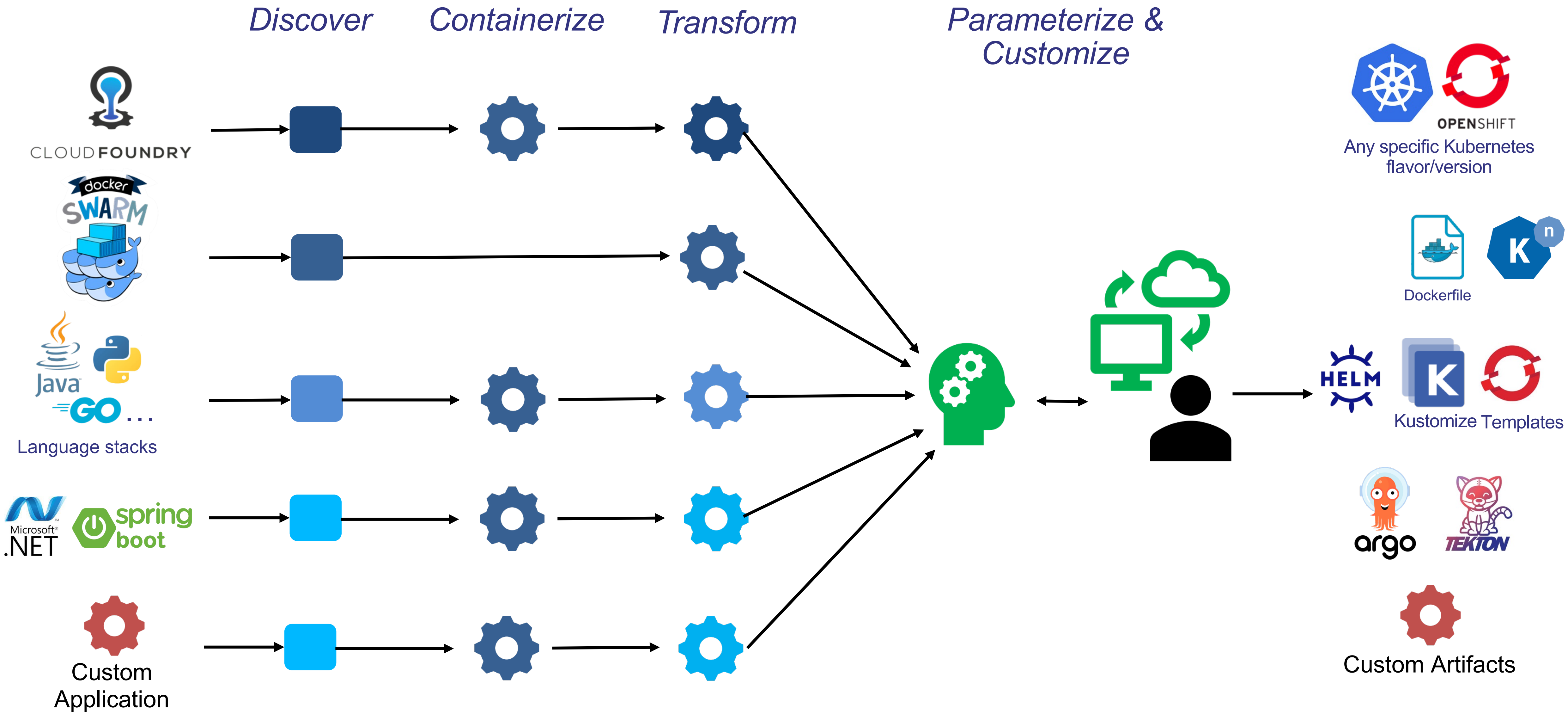
A CNCF Sandbox project

CLOUD NATIVE
COMPUTING FOUNDATION

KONVEYOR

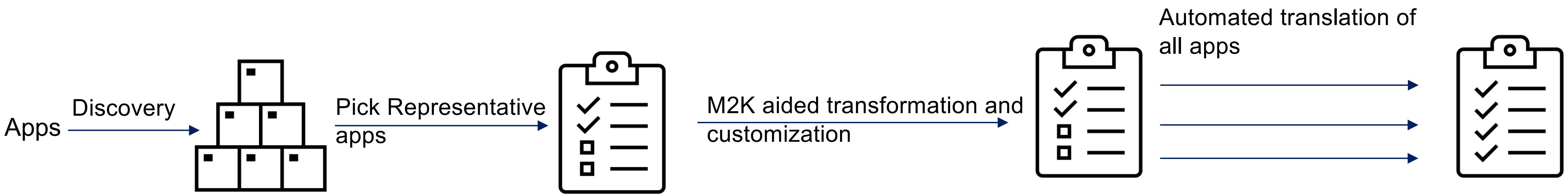
Konveyor Move2Kube

https://move2kube.konveyor.io/



*Move2Kube allows you to create all your Infrastructure as Code artifacts as per your organizational requirements. It allows **integrated discovery, containerization, transformation, parameterization and customization**.*

Konveyor Move2Kube Factory Approach



Apps

Discovery

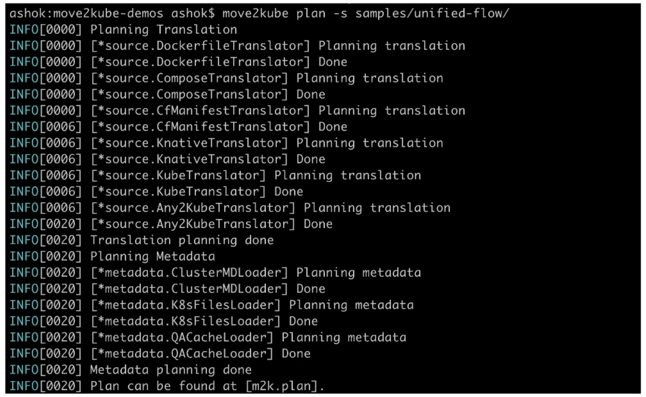
Pick Representative apps

M2K aided transformation and customization

Automated translation of all apps

Usage modes

Command line tool



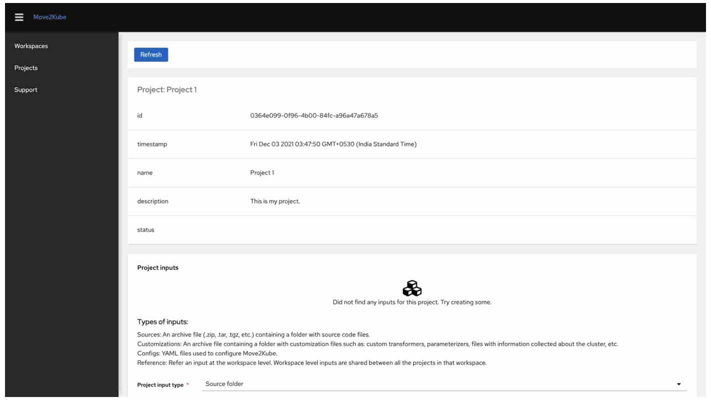
```
ashok:move2kube-demos ashok$ move2kube plan -s samples/unified-flow/
INFO[0000] Planning Translation
INFO[0000] [*source.DockerfileTranslator] Planning translation
INFO[0000] [*source.DockerfileTranslator] Done
INFO[0000] [*source.ComposeTranslator] Planning translation
INFO[0000] [*source.ComposeTranslator] Done
INFO[0000] [*source.CfManifestTranslator] Planning translation
INFO[0006] [*source.CfManifestTranslator] Done
INFO[0006] [*source.KnativeTranslator] Planning translation
INFO[0006] [*source.KnativeTranslator] Done
INFO[0006] [*source.KubeTranslator] Planning translation
INFO[0006] [*source.KubeTranslator] Done
INFO[0006] [*source.Any2KubeTranslator] Planning translation
INFO[0020] [*source.Any2KubeTranslator] Done
INFO[0020] Translation planning done
INFO[0020] Planning Metadata
INFO[0020] [*metadata.ClusterMDLoader] Planning metadata
INFO[0020] [*metadata.ClusterMDLoader] Done
INFO[0020] [*metadata.K8sFilesLoader] Planning metadata
INFO[0020] [*metadata.K8sFilesLoader] Done
INFO[0020] [*metadata.QACacheLoader] Planning metadata
INFO[0020] [*metadata.QACacheLoader] Done
INFO[0020] Metadata planning done
INFO[0020] Plan can be found at [m2k.plan].
```

```
bash <(curl https://raw.githubusercontent.com/konveyor/move2kube/main/scripts/install.sh)
```

GitHub
Releases

```
brew tap konveyor/move2kube
brew install move2kube
```

Web Interface



Project: Project 1

id	0364e099-0f96-4b00-84fc-a96a47a678a5
timestamp	Fri Dec 03 2021 03:47:50 GMT+0530 (India Standard Time)
name	Project 1
description	This is my project.
status	

Project inputs

Did not find any inputs for this project. Try creating some.

Types of inputs:
Sources: An archive file (.zip, .tar, .tgz, etc.) containing a folder with source code files.
Customizations: An archive file containing a folder with customization files such as: custom transformers, parameterizers, files with information collected about the cluster, etc.
Configs: YAML files used to configure Move2Kube.
Reference: Refer an input at the workspace level. Workspace level inputs are shared between all the projects in that workspace.

Project input type * Source folder

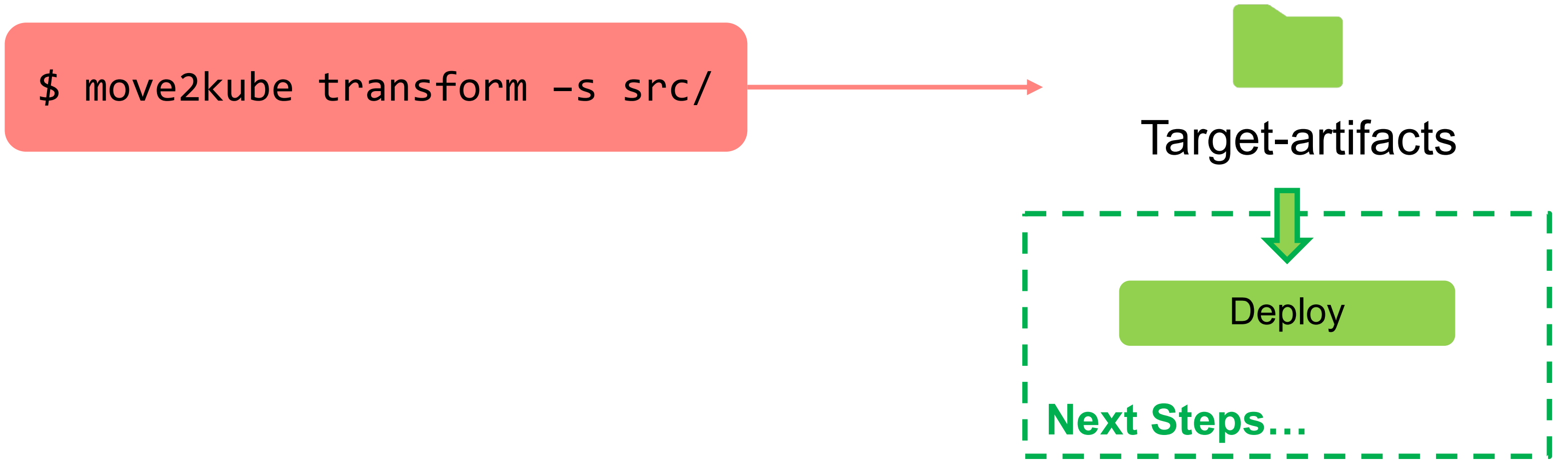
HELM

Operator

docker Compose

https://github.com/konveyor/move2kube

One step usage



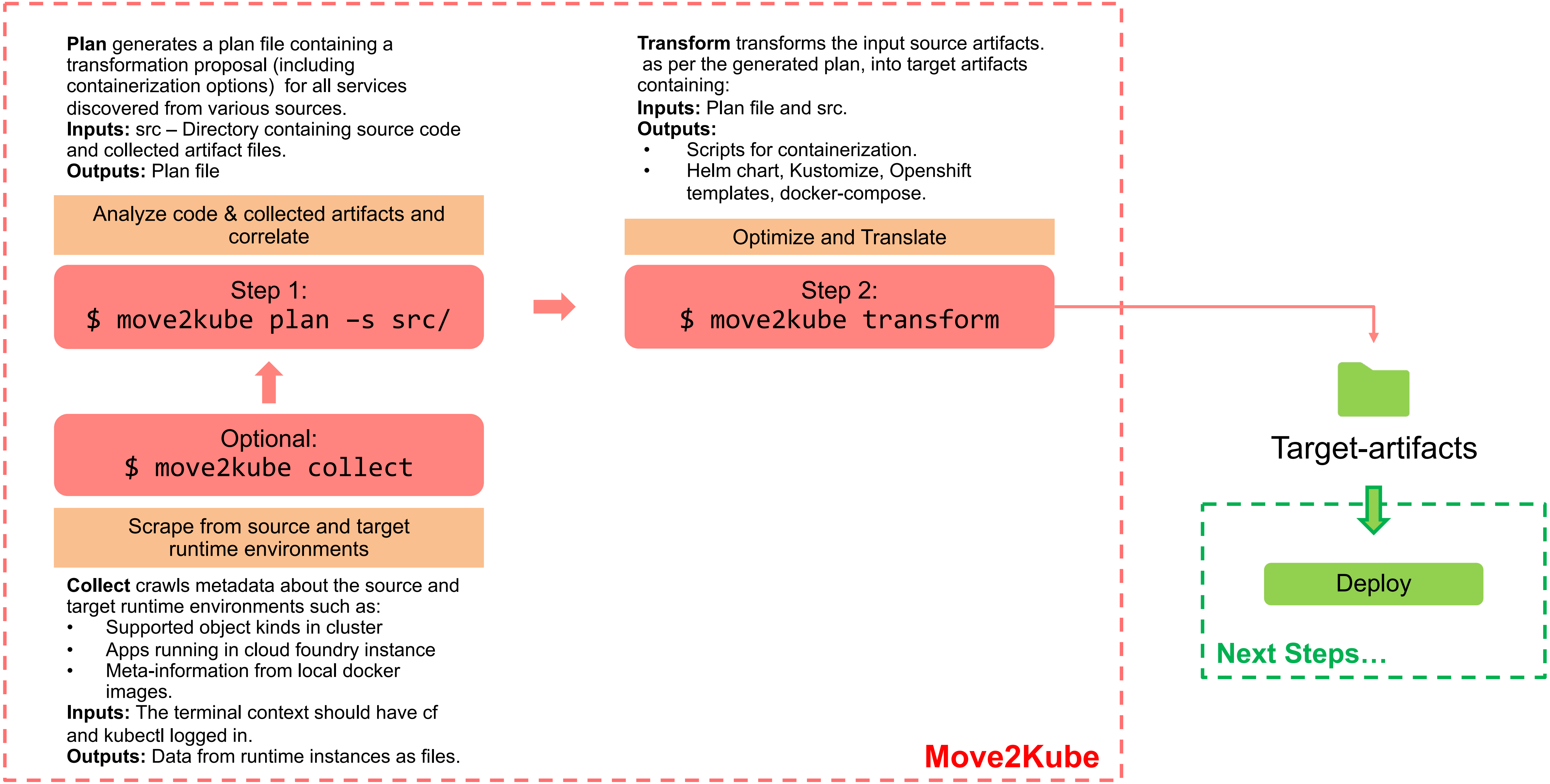
```
$ move2kube transform -s src/
```

Target-artifacts

Deploy

Next Steps…

Involved Usage



Plan generates a plan file containing a transformation proposal (including containerization options) for all services discovered from various sources.
Inputs: src – Directory containing source code and collected artifact files.
Outputs: Plan file

Analyze code & collected artifacts and correlate

Step 1:

```
$ move2kube plan -s src/
```

Optional:

```
$ move2kube collect
```

Scrape from source and target runtime environments

Collect crawls metadata about the source and target runtime environments such as:

- Supported object kinds in cluster
- Apps running in cloud foundry instance
- Meta-information from local docker images.

Inputs: The terminal context should have cf and kubectl logged in.
Outputs: Data from runtime instances as files.

Transform transforms the input source artifacts. as per the generated plan, into target artifacts containing:
Inputs: Plan file and src.
Outputs:

- Scripts for containerization.
- Helm chart, Kustomize, Openshift templates, docker-compose.

Optimize and Translate

Step 2:

```
$ move2kube transform
```

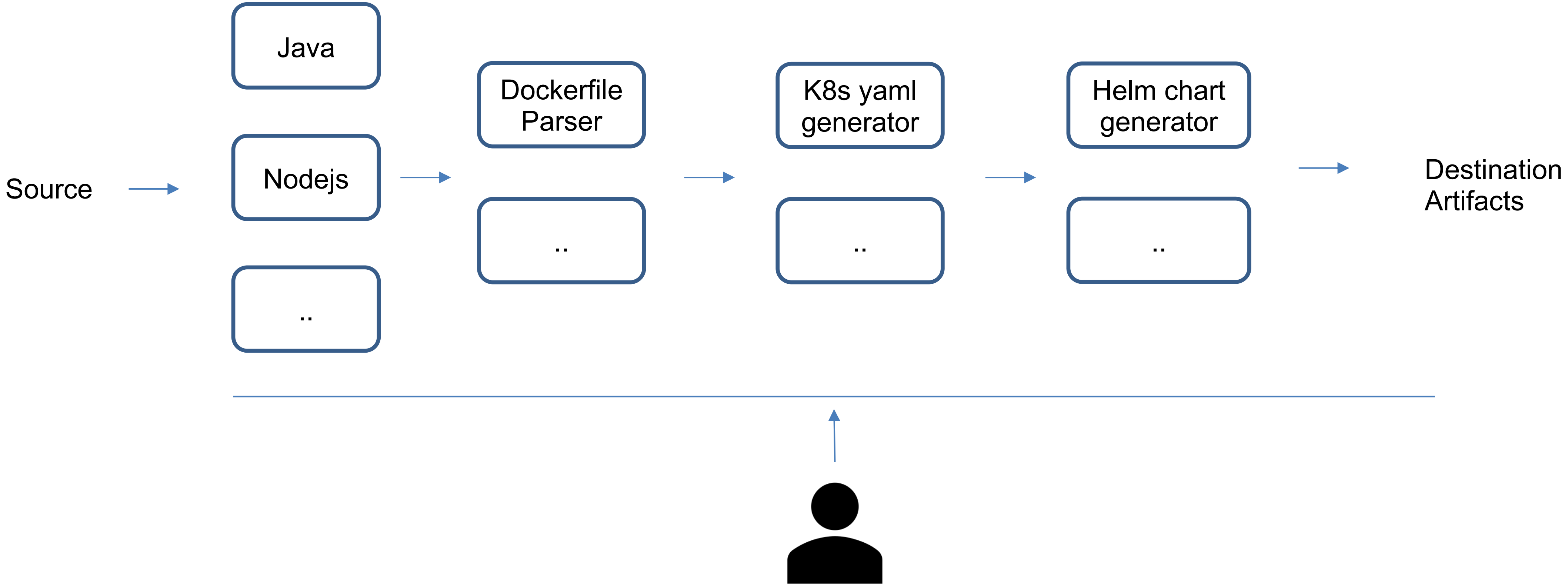
Target-artifacts

Deploy

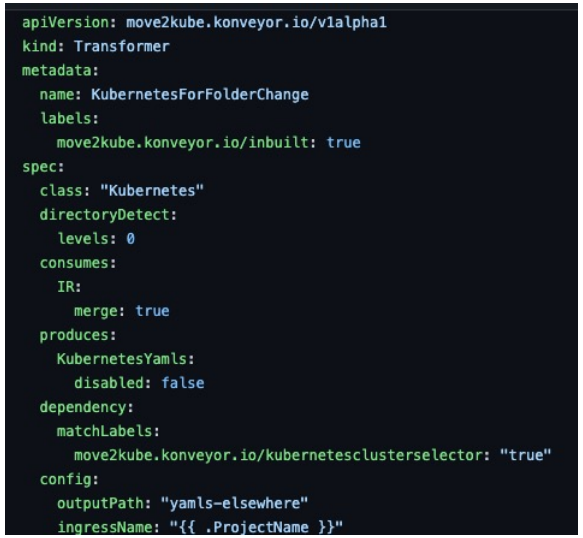
Next Steps…

Move2Kube

Design: Move2Kube Transformer Framework



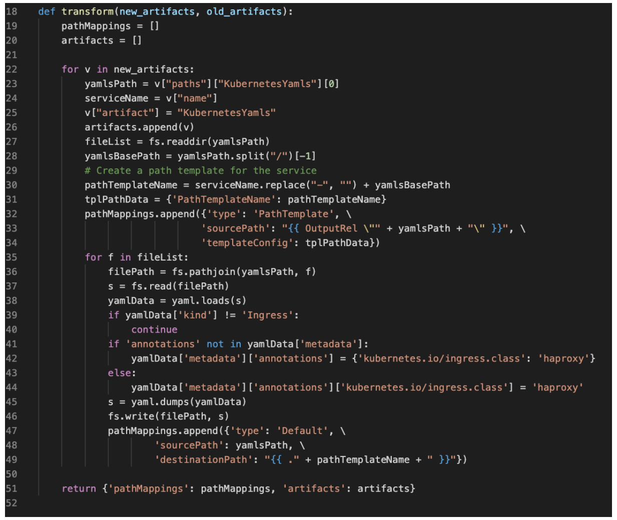
Customizing Transformers



```yaml
apiVersion: move2kube.konveyor.io/v1alpha1
kind: Transformer
metadata:
  name: KubernetesForFolderChange
  labels:
    move2kube.konveyor.io/inbuilt: true
spec:
  class: "Kubernetes"
  directoryDetect:
    levels: 0
  consumes:
    IR:
      merge: true
  produces:
    KubernetesYamls:
      disabled: false
  dependency:
    matchLabels:
      move2kube.konveyor.io/kubernetesclusterselector: "true"
  config:
    outputPath: "yamls-elsewhere"
    ingressName: "{{ .ProjectName }}"
```

Customizing in-built transformers

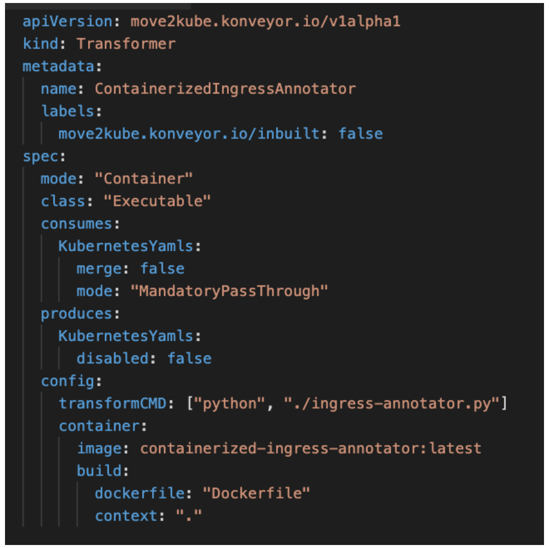
Each transformer exposes configurations which can be used to customize the transformer



```python
def transform(new_artifacts, old_artifacts):
    pathMappings = []
    artifacts = []

    for v in new_artifacts:
        yamlsPath = v["paths"]["KubernetesYamls"][0]
        serviceName = v["name"]
        v["artifact"] = "KubernetesYamls"
        artifacts.append(v)
        fileList = fs.readdir(yamlsPath)
        yamlsBasePath = yamlsPath.split("/")[-1]
        # Create a path template for the service
        pathTemplateName = serviceName.replace("-", "") + yamlsBasePath
        tplPathData = {'PathTemplateName': pathTemplateName}
        pathMappings.append({'type': 'PathTemplate', \
                            'sourcePath': "{{ OutputRel \"" + yamlsPath + "\" }}", \
                            'templateConfig': tplPathData})
        for f in fileList:
            filePath = fs.pathjoin(yamlsPath, f)
            s = fs.read(filePath)
            yamlData = yaml.loads(s)
            if yamlData['kind'] != 'Ingress':
                continue
            if 'annotations' not in yamlData['metadata']:
                yamlData['metadata']['annotations'] = {'kubernetes.io/ingress.class': 'haproxy'}
            else:
                yamlData['metadata']['annotations']['kubernetes.io/ingress.class'] = 'haproxy'
            s = yaml.dumps(yamlData)
            fs.write(filePath, s)
            pathMappings.append({'type': 'Default', \
                    'sourcePath': yamlsPath, \
                    'destinationPath': "{{ ." + pathTemplateName + " }}"})

    return {'pathMappings': pathMappings, 'artifacts': artifacts}

```

Custom transformers in starlark

A completely functional transformer can be written in starlark (python like)



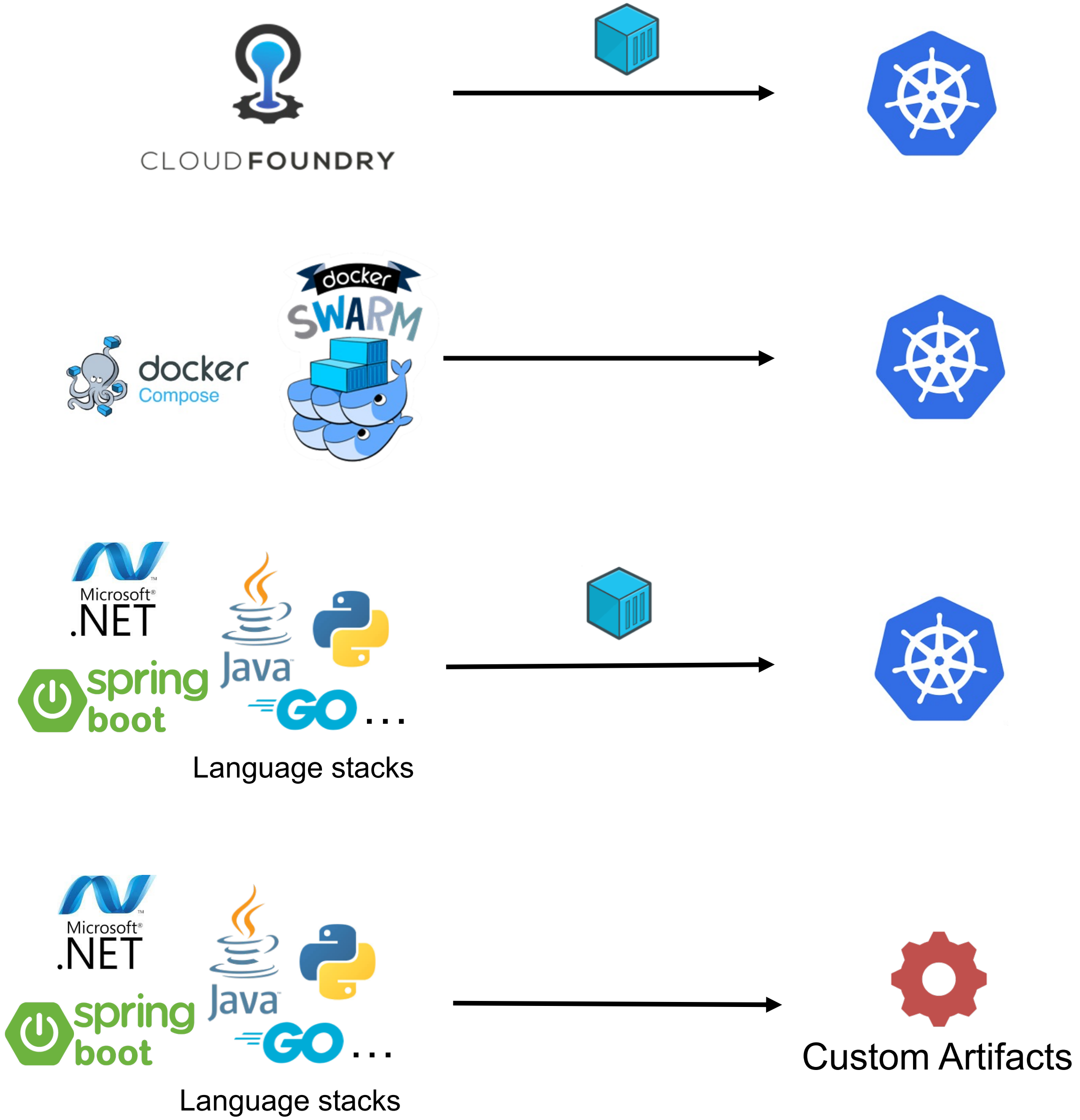
```yaml
apiVersion: move2kube.konveyor.io/v1alpha1
kind: Transformer
metadata:
  name: ContainerizedIngressAnnotator
  labels:
    move2kube.konveyor.io/inbuilt: false
spec:
  mode: "Container"
  class: "Executable"
  consumes:
    KubernetesYamls:
      merge: false
      mode: "MandatoryPassThrough"
  produces:
    KubernetesYamls:
      disabled: false
  config:
    transformCMD: ["python", "./ingress-annotator.py"]
    container:
      image: containerized-ingress-annotator:latest
      build:
        dockerfile: "Dockerfile"
        context: "."
```

Custom transformers as container

A completely functional transformer can be written in any language, and can be packaged as a container

https://github.com/konveyor/move2kube-transformers

Sample Usecases



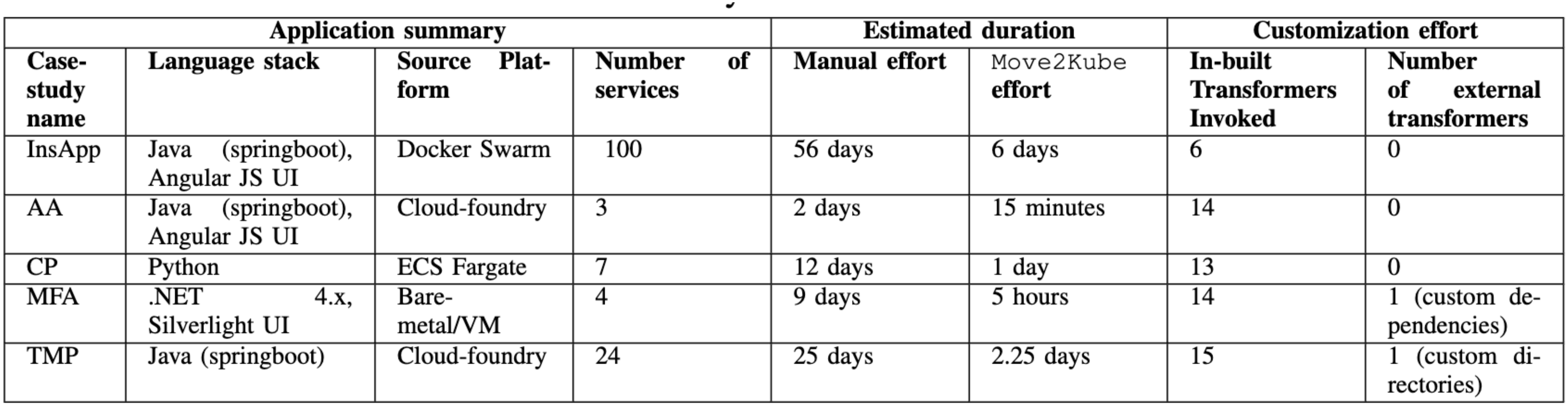
Language stacks

Language stacks

Custom Artifacts

https://move2kube.konveyor.io/tutorials

Summary of results for case-studies



Application summary				Estimated duration		Customization effort	
Case-study name	**Language stack**	**Source Platform**	**Number of services**	**Manual effort**	`Move2Kube` **effort**	**In-built Transformers Invoked**	**Number of external transformers**
InsApp	Java (springboot), Angular JS UI	Docker Swarm	100	56 days	6 days	6	0
AA	Java (springboot), Angular JS UI	Cloud-foundry	3	2 days	15 minutes	14	0
CP	Python	ECS Fargate	7	12 days	1 day	13	0
MFA	.NET 4.x, Silverlight UI	Bare-metal/VM	4	9 days	5 hours	14	1 (custom dependencies)
TMP	Java (springboot)	Cloud-foundry	24	25 days	2.25 days	15	1 (custom directories)