OpenCompose filespec

This document lists and describes all the elements of the proposed filespec for the standardised orchestration file aimed at developers. It also identifies mappings to existing orchestration solutions such as Docker Compose, Kubernetes/OpenShift and Apache Mesos.

File format

	Description	Value	Note
File name	The name of the file created by a developer that follows this filespec	<choreography>.yaml	The name is to be confirmed. We are currently using a placeholder (<choreorgraphy>).
File format	The format of the file created by a developer that follows this filespec	YAML	See http://yaml.org/ for a full specification of the yaml format
Version	The version of this filespec	0.1	This is currently a pre-alpha proposal and under heavy development

Element Structure

```
version: "0.1"
services:
  <service name>:
    container_name: <String>
    context: <Path>
    dockerfile: <Path>
    command: <String>
    entrypoint: <String>
    image: <String>
    extends:
      file: <Path>
      service: <Reference>
    depends_on:
      - <Service Reference>
      - <Service Reference>
    environment:
      - <Key=Value>
      - <Key=Value>
    env_file: <String>
    env_file:
       - <Path>
       - <Path>
    expose:
       - <Container>
    ports:
       - <Host:Container>
    labels:
       - <Label: Value>
    links:
       - <Reference>
```

```
      - <Reference>
    volumes:
      - <Host:Container>
    volumes_from:
      - <Service Reference>
```

Example

The Helloworld MSA project serves as an example. It contains at least 10 Openshift Applications (6 microservices, api-gateway, frontend, kubeflix, zipkin).

Github URL: https://github.com/redhat-helloworld-msa/helloworld-msa

TODO Add <choreography> support to MSA example

Element Overview

Object	Description	Marathon Object	Compose Object	Kubernetes Object
build	Builds a container using a linux container		build	**BuildConfig**
context	Either a path to a directory containing a *Dockerfile*, or a url to a git repository		context	**BuildConfig**.spec.source.contextDir
dockerfile	Alternate *Dockerfile*.		dockerfile	N/A - This is the default file used by the

				BuildConfig/Docker strategy
command	Override the default command.		command	**Pod**.spec.container.command
container_name	Specify a custom container name, rather than a generated default name.		container_name	**ObjectMeta**/name
depends_on	Express dependency between services		depends_on	Directed service dependencies (coming soon)
environment	Add environment variables		environment	**Container/**env
expose	Expose ports without publishing them to the host machine		expose	**Service**
extends	Extend another service, in the current file or another, optionally overriding configuration.		extends	N/A
image	Specify the image to start the container from.		image	**BuildConfig**/image
labels	Add metadata to containers using Docker labels		labels	**ObjectMeta**/annotations
net	Network mode. Use the same values as the docker client --net parameter		net	N/A - not needed

ports	Expose ports.		ports	**Service**[1]
Volumes	Mount paths or named volumes		volumes	**DeploymentConfig**/volume[2]
volumes_from	Mount all of the volumes from another service or container		volumes_from	N/A - use volumes instead Colocation + volume mounts of all the defined mounts on the first image.
entrypoint	Override the default entrypoint.		entrypoint	**Pod**.spec.container.command
env_file	Add environment variables from a file. Can be a single value or a list.		env_file	N/A
links	Link to containers in another service. Either specify both the service name and a link alias (SERVICE:ALIAS), or just the service name.		links	Pod.spec.container

Mapping exists	Mapping not required	Problematic

[1] A non-trivial number of docker compose files support remapping ports - and so pod -> service relationships involve three entities (pod port, service port -> pod port, and service external port)
[2] Volumes, as defined in a compose file, can express implicit colocation into a pod. So the act of sharing a volume mount is == being in a pod.

Detailed ElementSpec

TODO
Until this section is complete, please refer to the identically named elements from the docker compose file reference at https://docs.docker.com/compose/

Open Questions

Open Questions
How do we allow the developer to express that services should be colocated? Comments: Network info / Volume Sharing / links is currently used to express this in Docker compose. Everything or nothing
How do we allow the developer express minimum capabilities required to run (e.g. disk space, memory, CPU)?
How do we allow implementations to provide inline or separate file extensions to the descriptor, so that they can innovate and add capabilities? (ideally without breaking the descriptor)
Handle replicas
Portability to windows - link to resources/capabilities
Multi-file bundle/ or overrrides (CSS style)
Specify systems availability?