DOCKER-COMPOSE TIPS

INDEX.JS

- Validation
- Environment Variables
- Multiple compose files
- External Networks
- Commands
- Anchors

VALIDATION

docker-compose config validates and outputs configuration.
It's essential in more advanced usages, as we'll see.

DC CONFIG - ERROR

```yaml
# docker-compose.yml
version: '3.3'

service:  # Typo here
  app:
    image: nginx/alpine
```

```
$ docker-compose config

ERROR: The Compose file is invalid because:

Invalid top-level property "service".
Valid top-level sections for this Compose file are:
  version, services, networks, volumes, secrets, configs,
  and extensions starting with "x-".
```

Spoiler alert: extensions

DC CONFIG - GOOD BOY

```yaml
# docker-compose.yml
version: '3.3'

services:
  app:
    image: nginx/alpine
```

```
$ docker-compose config

services:
  app:
        image: nginx/alpine
version: '3.3'
```

Side note: changed order of yml sections

ENVIRONMENT VARIABLES

docker-compose can make use of environment variables to be more configurable and dynamic.

ENVIRONMENT VARIABLES - EXAMPLE

```yaml
# docker-compose.yml
version: '3.3'

services:
  app:
    image: my-registry/my-docker-image:${TAG}
```

```
$ export TAG=v3.1.0
$ docker-compose config

services:
  app:
        image: my-registry/my-docker-image:v3.1.0
version: '3.3'
```

Defaults could be provided with the syntax:
${VARIABLE:-default} or ${VARIABLE:default}

.ENV SPECIAL FILE

```
# .env
TAG=v3.2.0
```

```
# docker-compose.yml
version: '3.3'

services:
  app:
    image: my-registry/my-docker-image:${TAG}
```

```
$ docker-compose config

services:
  app:
        image: my-registry/my-docker-image:v3.2.0
version: '3.3'
```

If a **.env** file exists, it'll be used as source for env vars.

MULTIPLE FILES

docker-compose can accepts more than one file, overriding values.
This behaviour augments exponentially the compose capabilities

MULTIPLE FILES - EXAMPLE

```
# docker-compose.yml
version: "3.3"
services:
  app:
    image: nginx/alpine
```

```
# docker-compose.prod.yml
version: "3.3"
services:
  app:
    volumes:
          - "./config/:/app/config/"
```

```
$ docker-compose -f docker-compose.yml -f docker-compose.prod.yml config
services:
  app:
    image: nginx/alpine
    volumes:
      - /absolute/path/calculated/config:/app/config:rw
version: '3.3'
```

MULTIPLE FILES - FAIL

From great powers ... come greater fails

MULTIPLE FILES - FAIL

```yaml
# docker-compose.yml
version: "3.3"
services:
  app:
    image: nginx/alpine
    ports:
        - "8080:8080"
```

```yaml
# docker-compose.prod.yml
version: "3.3"
services:
  app:
    ports:
      - "8181:8080"
```

```
$ docker-compose -f docker-compose.yml -f docker-compose.prod.yml config
```

What will be the rendered **ports** property?

```yaml
services:
  app:
    image: nginx/alpine
    ports:
      - "8080:8080"
      - "8181:8080"
version: '3.3'
```

YAML lists are appended, not overridden

MULTIPLE FILES - THE CORRECT WAY

```
# docker-compose.local.yml
version: "3.3"
services:
  app:
    ports:
        - "8080:8080"
```

```
# docker-compose.prod.yml
version: "3.3"
services:
  app:
    ports:
        - "8181:8080"
```

```
# docker-compose.yml
version: "3.3"
services:
  app:
    image: my-registry/my-image:$TAG
```

```
$ TAG=v6.6.6 docker-compose -f docker-compose.yml -f docker-compose.prod.yml config
services:
  app:
    image: my-registry/my-image:v6.6.6
    ports:
        - "8181:8080"
version: '3.3'
```

MULTIPLE FILES - VIA ENV VAR

It's possible to use an environment variable to specify compose files.

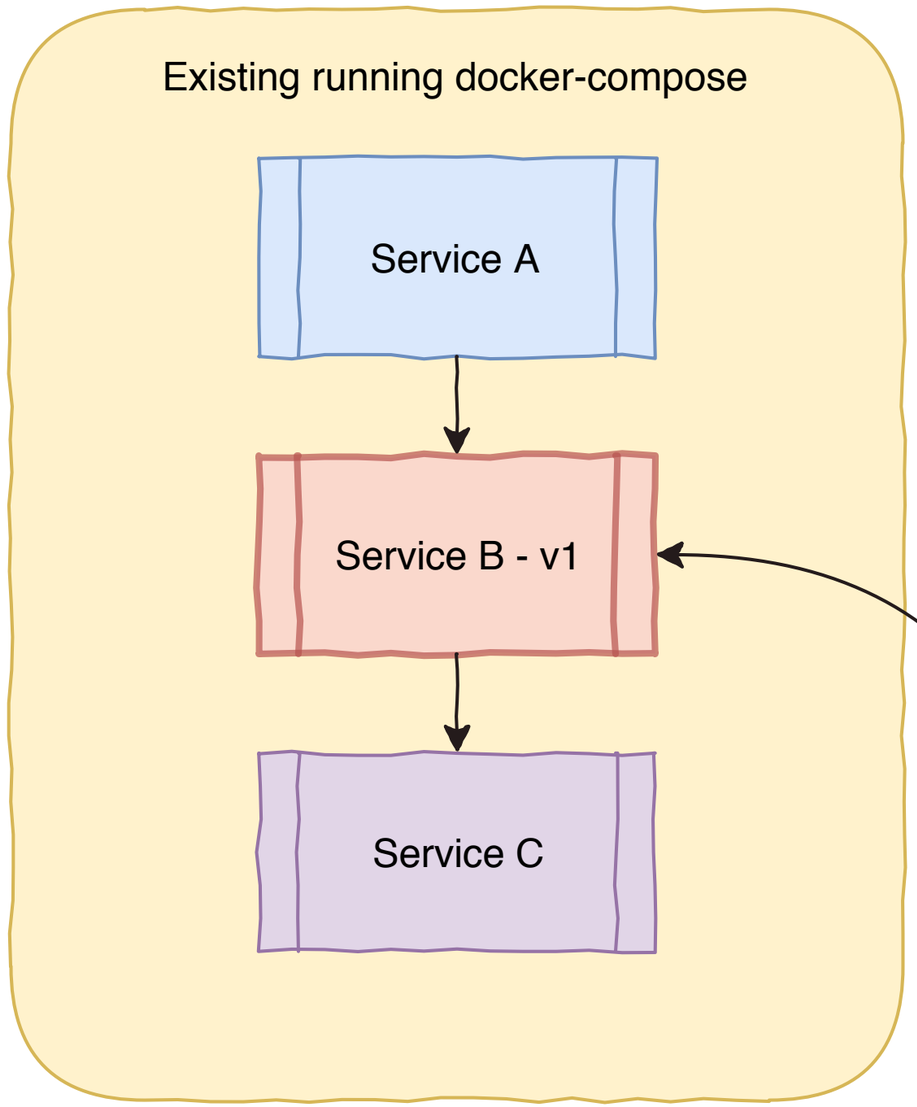
```
# .env
COMPOSE_FILE=docker-compose.yml:docker-compose.local.yml
# ":" separated string
TAG=latest
```

```
$ docker-compose config
services:
  app:
    image: my-registry/my-image:latest
    ports:
        - "8080:8080"
version: '3.3'
```

NETWORKING

docker-compose networking is flexible, and permits to use existing (external) networks, and also control how the internal DNS will behave

ATTACHING TO OTHER NETWORKS - USE CASE



Use Case

The service you are developing is a gateway between upstream and downstream services.

It would be handy for development to use an existing compose substituting your local version with the one running

ATTACHING TO OTHER NETWORKS - COMPOSE

```yaml
# docker-compose.serviceB.yml
version: '3.7'
services:
  serviceB:
    build: { "context": "." }
    networks:
      otherComposeNetwork:
        aliases:
          - ${NETWORK_SERVICE_ALIAS:-serviceB}
networks:
  otherComposeNetwork:
    external:
      name: ${NETWORK_NAME:-crystal_framework}
```

```
$ docker network list
NETWORK ID          NAME                      DRIVER              SCOPE
b753eefc8242        somecompose_network       bridge              local

$ docker-compose -f other-compose.yml stop serviceName

$ export NETWORK_SERVICE_ALIAS=serviceName
$ export NETWORK_NAME=somecompose_network
$ docker-compose up
```

Et voila'!

COMMANDS

It's possible to exec into running containers.

Very useful for running integration/e2e tests!

```
$ docker-compose exec serviceName npm integration
```

ANCHORS

Docker-compose will ignore "x-" keys in validation, in that way is possible to use normal YAML anchors.

```yaml
version: '3.4'
x-logging:
  &default-logging
  options:
    max-size: '12m'
    max-file: '5'
  driver: json-file

services:
  web:
    image: myapp/web:latest
    logging: *default-logging
  db:
    image: mysql:latest
    logging: *default-logging
```

```
$ docker-compose config
services:
  db:
    image: mysql:latest
    logging:
      driver: json-file
      options:
        max-file: '5'
        max-size: 12m
  web:
    image: myapp/web:latest
    logging:
      driver: json-file
      options:
        max-file: '5'
        max-size: 12m
version: '3.4'
```

USEFUL LINKS

- Environment variables
- CLI Environment variables
- Env file
- Multiple files and "extends"
- Startup order
- Networking

Q & A ?

THANKS! BYE! 👋