Nulecule

Packaging, Distributing & Deploying Container Applications the Cloud Way

2016 - Ghent, Belgium

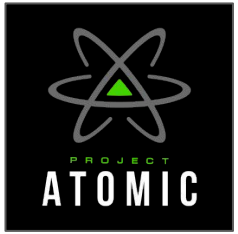
Brian Exelbierd

- Container Tools Engineer @ Red Hat

```
vpavlin@localhost $ su - bexelbie
bexelbie@localhost $
```

https://github.com/bexelbie
https://twitter.com/bexelbie

Slides heavily based on Václav Pavlín's Originals



Atomic Developer Bundle

An easy start Linux container development environment.

Enabling development with Docker, Kubernetes, OpenShift, Mesos-Marathon and Nulecule

Toolbox

<insert containers talk>

I don’t have to do this, do I?

Container Packaging

Simple, Clean & Beautiful*

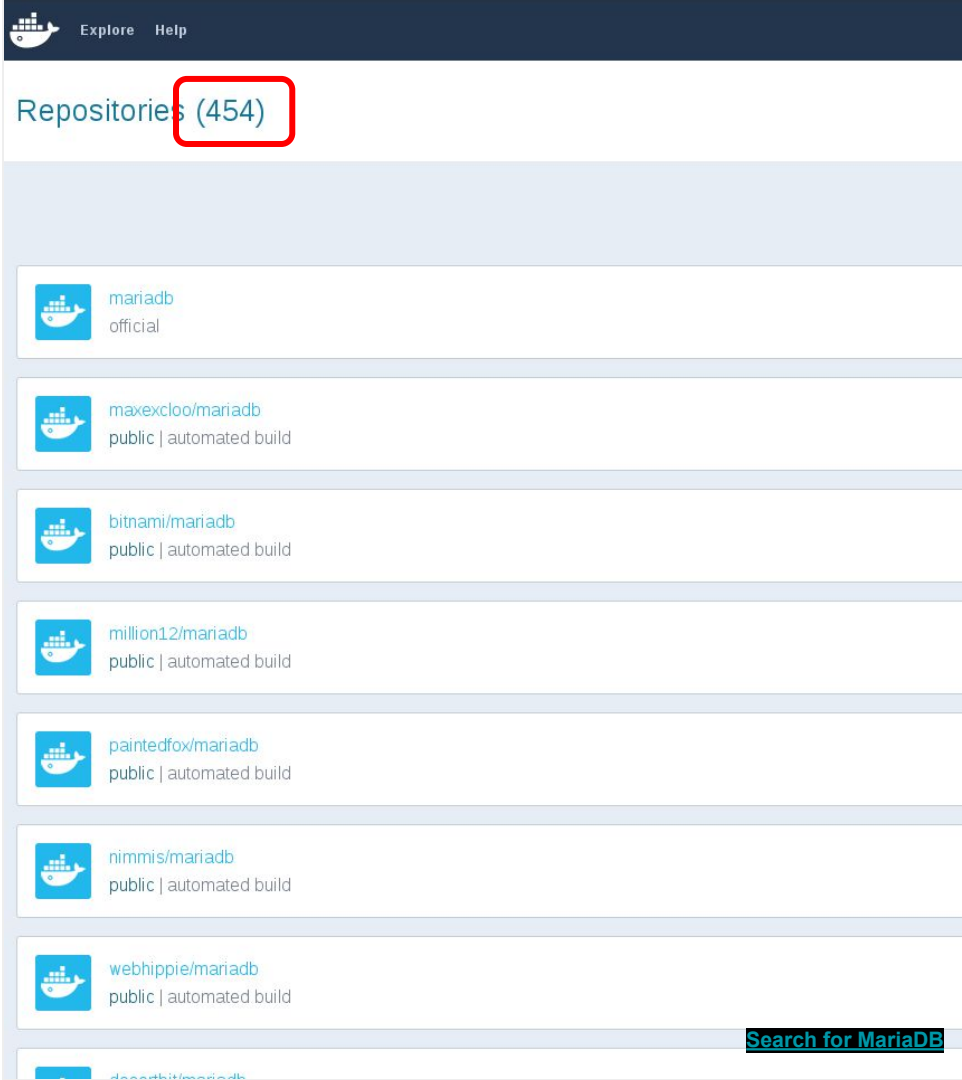
But … no dependency defs, no instructions, all “open” differently and everyone makes a new one.

*The debate on these terms is another talk

Beverage Can

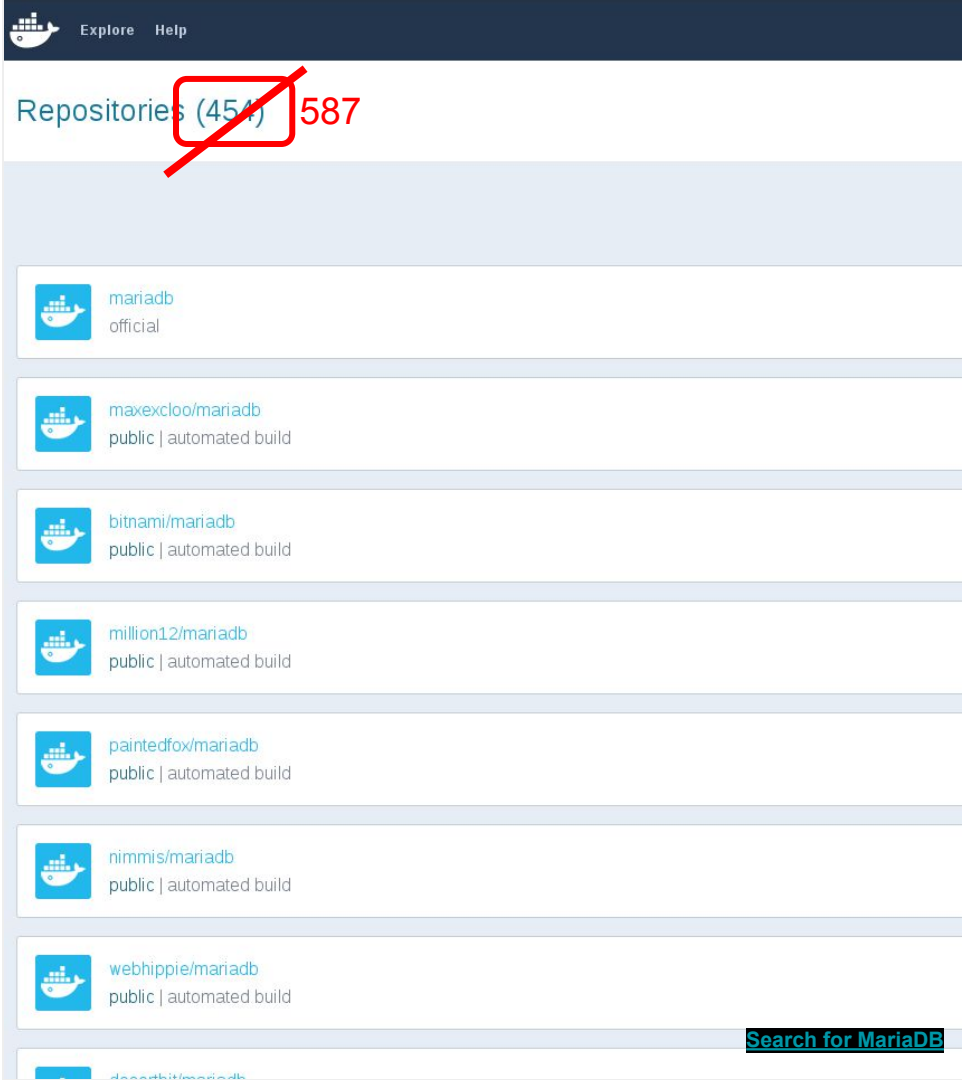
Everybody Repackages

Bonus: Most are poorly documented, not easily changed, not audited, and generally scary

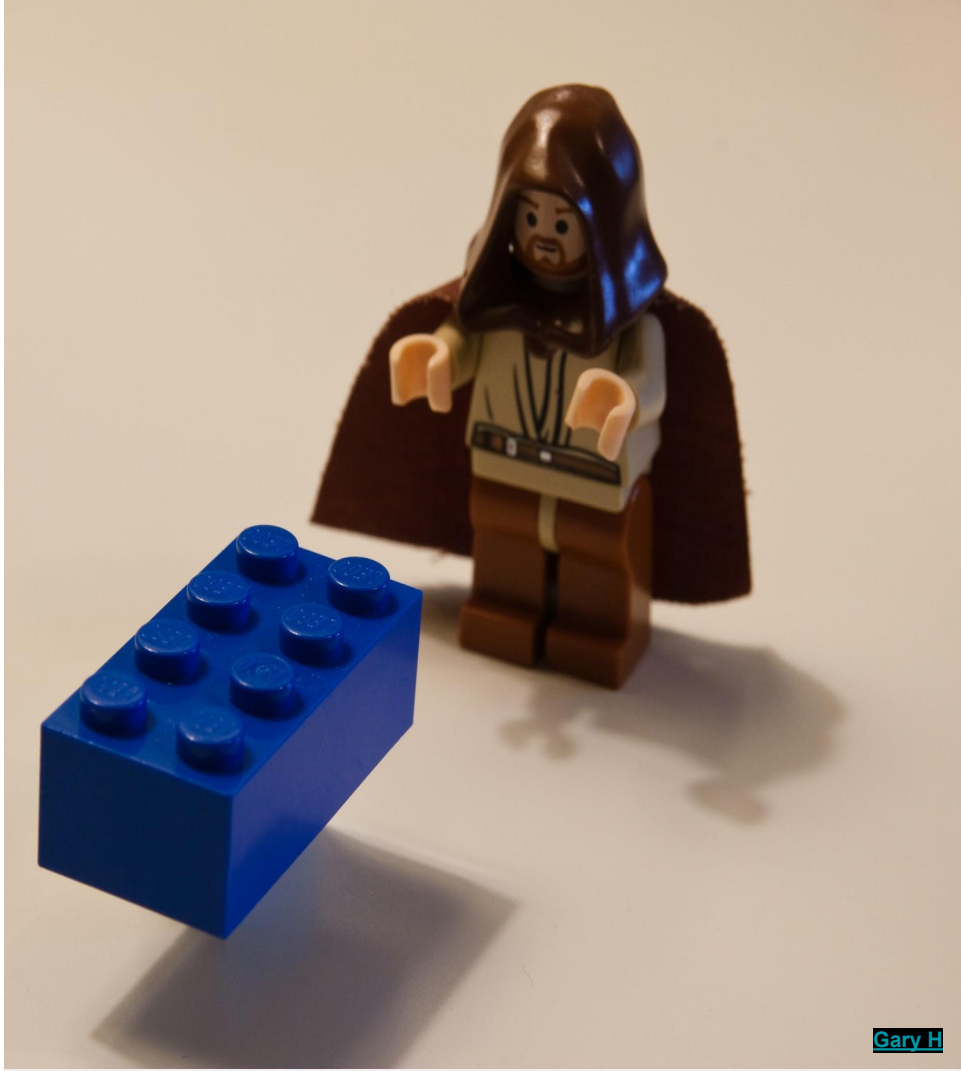


Everybody Repackages

Bonus: Most are poorly documented, not easily changed, not audited, and generally scary



Containers are fun!



READMEs

The “UX” of choice for containers

Run the mariadb container:

```
# docker run --name=mydb -e USER=wordpress -e
PASS=$(pwgen -s -1) -e NAME=wordpress -d
<yourname>/mariadb
```

Then run the wordpress container, using the alias 'db' for the linked MariaDB container:

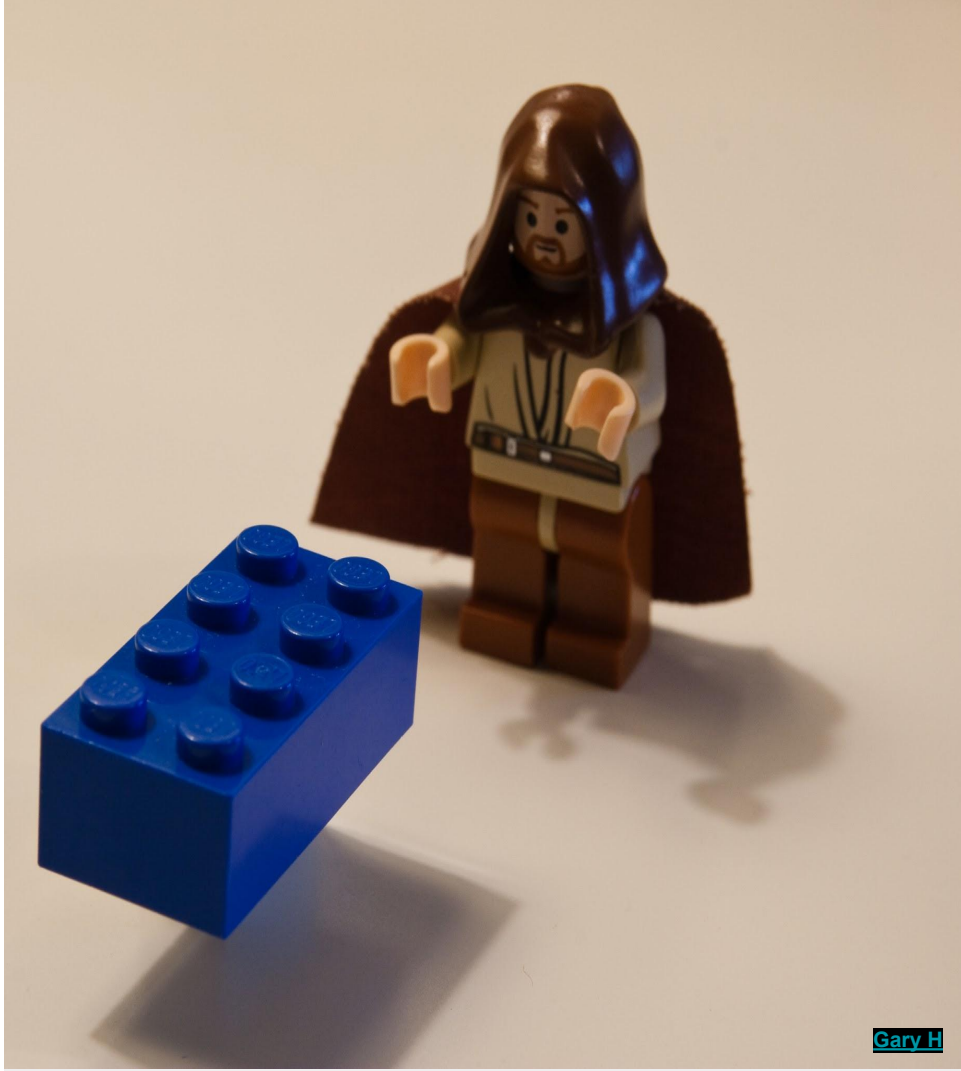
```
# CID=$(docker run -p 80 --link=mydb:db -d
<yourname>/wordpress)
```

Then find the external port assigned to your container:

```
# docker port $CID 80
```

Visit in a web browser, then fill out the form. No need to mess with wp-config.php, it has been auto-generated with proper values.

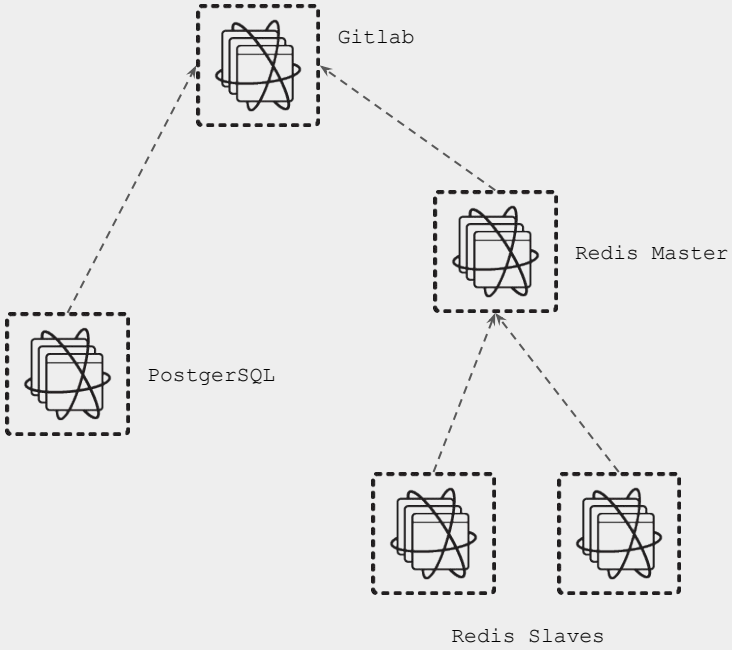
Containers are fun!



Multi-container Application

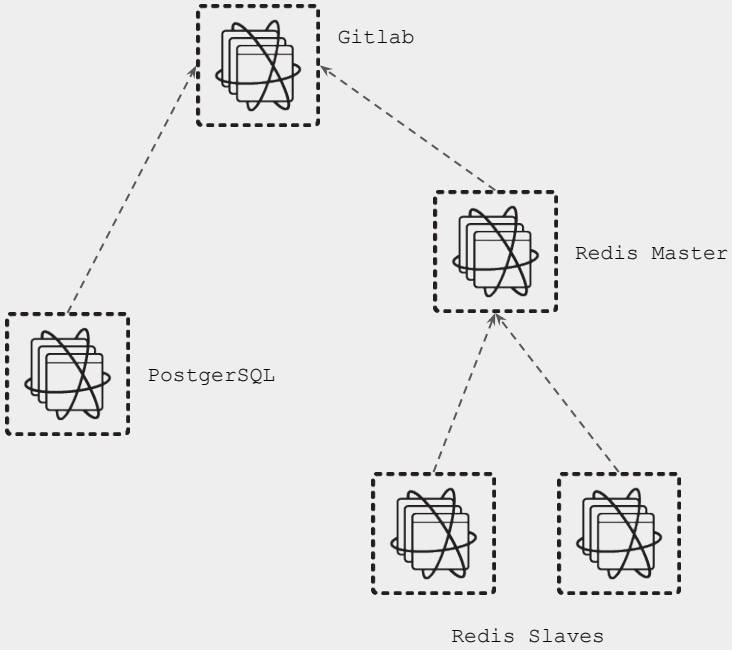
2-n container images, operated as a single unit, re-using existing components

An application consists of many parts, that need to be operated together



Coming Soon: Microservices

An application consists of many parts, that need to be operated together



Orchestration

Hooray - metadata separated from images

Metadata Distribution

No common way to transfer metadata to Ops or other users

```
$ curl -O https://raw.githubusercontent.com/kube...
$ ls
redis-master-controller.yaml
$ kubectl create -f redis-master-controller.yaml
```

Various Orchestration Projects

There is no winner yet and each defines it's own format to describe the deployment

Metadata Modifications

Most environment changes will require some metadata changes

```
"env": [
  {
    "name": "MYSQL_SERVICE_IP",
    "value": "1.2.3.4"
  },
  {
    "name": "MYSQL_SERVICE_PORT",
    "value": "3306"
  },
  {
    "name": "MYSQL_PASSWORD",
    "value": "1234"
  }
]
```

Note: Remember to substitute environment variable values in json file before creating replication controller.

Quoted from [Phabricator Kubernetes example]

READMEs

The “UX” of choice for multi-container orchestrated apps

Kubernetes Guestbook Example

Guestbook Example

This example shows how to build a simple, multi-tier web application using Kubernetes and Docker.

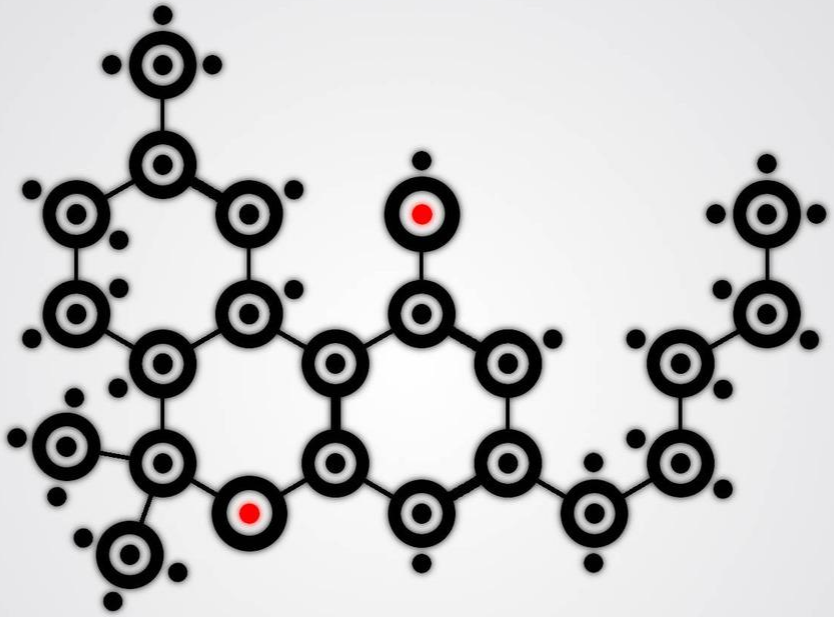
Table of Contents

The example consists of:

725 lines/paragraphs - +30 KB

Nulecule

Specification, composability, common distribution, parametrization, orchestration providers



Molecule Wallpaper

Just a Spec

Container engine independent
Orchestrator embracing

Docker
nspawn
lxd rkt
lxc

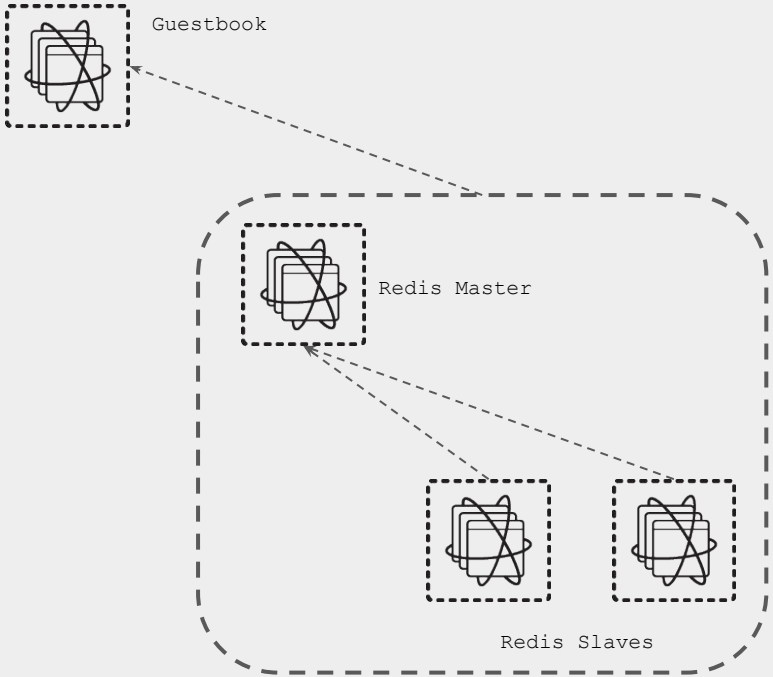
Why another thing?

- Tool agnostic - and doesn't push
- Allow high-level thought with low-level tweaks
- Easy enough for a junior sysadmin to use
- Able to integrate with existing tools
- Open, including implementatoin

Graph

DAG to define application components and dependencies

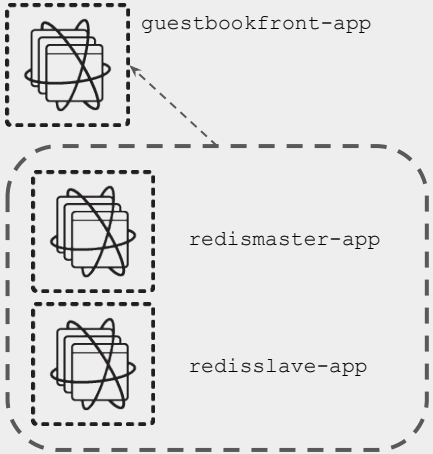
```
graph:
  - name: guestbookfront-app
    ...
  - name: redis-centos7-atomicapp
    ...
```



Manages Dependencies

All dependencies are pulled during “install” as defined in Nulecule.

```
graph:
  - name: guestbookfront-app
    ...
  - name: redis-centos7-atomicapp
    source: docker://projectatomic/redis-centos7-atomicapp
```




```
docker pull projectatomic/redis-centos7-atomicapp

graph:
  - name: redismaster-app
    ...
  - name: redisslave-app
    ...
```

Parametrization

Provide the right values at deployment time

```
graph:
  - name: helloapache-app
    params:
      - name: image
        description: The webserver image
        default: centos/httpd
      - name: hostport
        description: The host TCP port
        default: 80
        constraints:
          - allowed_pattern: ^[0-9]+$
            description: Port number has to be a
numeric value
```

Parametrization

Every component has its own parameters

```
graph:
  - name: helloapache-app
    params:
      - name: image
        description: The webserver image
        default: centos/httpd
      - name: hostport
        description: The host TCP port
        default: 80
        constraints:
          - allowed_pattern: ^[0-9]+$
            description: Port number has to be a
numeric value
```

Parametrization

Default values can be provided and overridden

```
graph:
  - name: helloapache-app
    params:
      - name: image
        description: The webserver image
        default: centos/httpd
      - name: hostport
        description: The host TCP port
        default: 80
        constraints:
          - allowed_pattern: ^[0-9]+$
            description: Port number has to be a
numeric value
```

Parametrization

Parameters can be constrained by regular expression

```
graph:
  - name: helloapache-app
    params:
      - name: image
        description: The webserver image
        default: centos/httpd
      - name: hostport
        description: The host TCP port
        default: 80
        constraints:
          - allowed_pattern: ^[0-9]+$
            description: Port number has to be a
numeric value
```

Answers file

A file containing “answers” to questions defined by parameters

```
[general]
provider = kubernetes

[helloapache-app]
image = centos/httpd
hostport = 80
```

Answers file

A file containing “answers” to questions defined by parameters

```
[general]
provider = kubernetes

[helloapache-app]
image = fedora/httpd
hostport = 8080
```

Providers

These represent orchestrators

```
artifacts:
  kubernetes:
    - file://...kubes/gitlab-rc.json
    - file://...kubes//gitlab-http-service.json
  docker:
    - file://...docker/gitlab-link-run
  openshift:
    - file://...shift/os-route.json
    - inherit:
      - kubernetes
```

Artifacts

Deployment metadata templates for orchestrators

```
artifacts:
  kubernetes:
    - file://...kubes/gitlab-rc.json
    - file://...kubes//gitlab-http-service.json
  docker:
    - file://...docker/gitlab-link-run
  openshift:
    - file://...shift/os-route.json
    - inherit:
      - kubernetes
```

Artifacts are Parameterized

Dollar sign variable replacement

```
"image": "$image",
"name": "helloapache",
"ports": [
    {
        "containerPort": 80,
        "hostPort": $hostport,
        "protocol": "TCP"
    }
]
```

It’s not Slideware …

Atomic App

- Reference Implementation of Nulecule
- Nulecule app installer and manager, container-enabled, provider plugins, single command deployment

Ship Shipping Ships

Base for application images

You build your app on top of our Atomic App base image

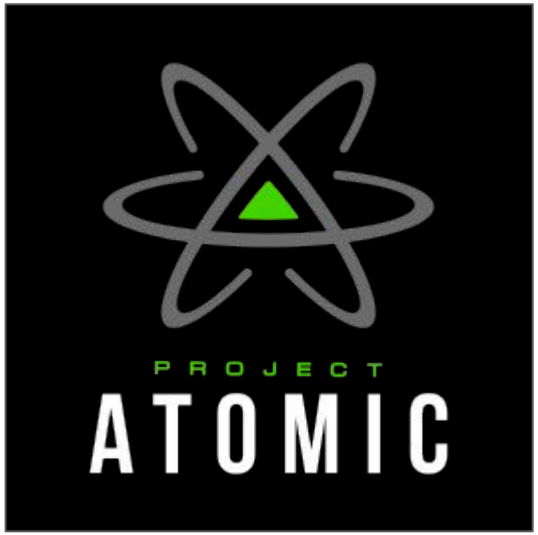
```
FROM projectatomic/atomicapp:0.4.0

MAINTAINER Red Hat, Inc. <container-tools@redhat.com>

LABEL io.projectatomic.nulecule.specversion="0.0.2" \
      io.projectatomic.nulecule
            providers="kubernetes,docker"\
      Build="docker build --rm --tag
            test/gitlab-atomicapp ."

ADD /Nulecule /Dockerfile README.md gpl-3.0.txt
      /application-entity/

ADD /artifacts /application-entity/artifacts
```

Demo Thanks:
Tomas Kral (@kadel)
Michael Scherer

DEMO TIME

Indiana Jones and the Temple of Doom

Learn More at



- Project Site: www.projectatomic.io
- Github:
  - https://github.com/projectatomic/nulecule
  - https://github.com/projectatomic/atomicapp
- IRC: #nulecule @ Freenode
- Mailing List: container-tools@redhat.com

Brian “bex” Exelbierd @bexelbie, bex@pobox.com, bexelbie@redhat.com

Slides and examples: https://github.com/bexelbie/nulecule-talk-demo