Nulecule

Packaging, Distributing & Deploying Container Applications the Cloud Way
Brian Exelbierd

- Container Tools Engineer @ Red Hat

 vpavlin@localhost $ su - bexelbie
 bexelbie@localhost $
Atomic Developer Bundle

An easy start Linux container development environment.

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

*Beverage Can

*The debate on these terms is another talk
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!
Run the mariadb container:

```bash
# 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:

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

Then find the external port assigned to your container:

```bash
# 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
An application consists of many parts, that need to be operated together.

Coming Soon: Microservices
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 its own format to describe the deployment.

Projects:
- OpenShift
- Flynn
- Compose
- Mesos+Marathon
- Kubernetes
- Dokku
- ShutIt
- Terraform
- Helios
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

Guestbook Example

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

Table of Contents

- Guestbook Example
  - Prerequisites
  - Quick Start
    - Step One: Start up the redis master
      - Define a replication controller
      - Define a service
      - Create a service
      - Finding a service
      - Create a replication controller
      - Optional Interlude
    - Step Two: Start up the redis slave
    - Step Three: Start up the guestbook frontend
      - Using ‘type: LoadBalancer’ for the frontend service (cloud-provider-specific)
    - Step Four: Cleanup
  - Troubleshooting
  - Appendix: Accessing the guestbook site externally
    - Google Compute Engine External Load Balancer Specifics

The example consists of:
Nulecule

Specification, composability, common distribution, parametrization, orchestration providers
Just a Spec

Container engine independent
Orchestrator embracing

Docker
nspsawn
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 implementation
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.
Parametrization

Provide the right values at deployment time

```yaml
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

```json
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]+$
Parametrization

Default values can be provided and overridden

```yaml
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

```yaml
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]+$  # 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

[generic]
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
It’s not Slideware …

Atomic App

- Reference Implementation of Nulecule
- Nulecule app installer and manager, container-enabled, provider plugins, single command deployment
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"\n    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
Learn More at

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

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

Slides and examples: [https://github.com/bexelbie/nulecule-talk-demo](https://github.com/bexelbie/nulecule-talk-demo)