jahia

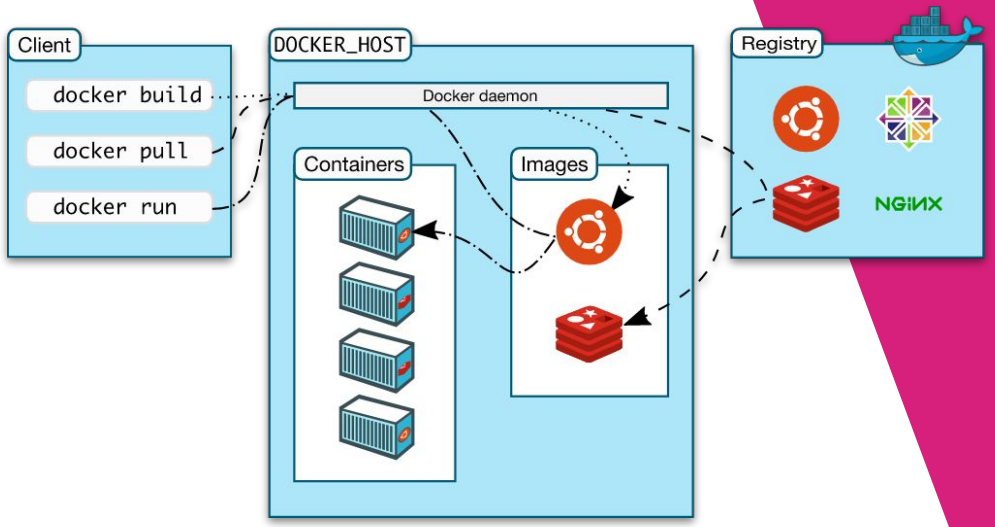
Automate your Jahia deployments with Docker & our provisioning API

Serge Huber, Jahia CTO

shuber@jahia.com

What is Docker ?

- An Open-Source containerization platform
- It packages an application and all its runtime dependencies into a standardized container format.
- As opposed to Virtual Machines, Docker containers do not emulate the underlying hardware, providing greater performance and portability.
- Docker images can be executed on Linux servers using the Docker engine, Cloud services such as Amazon ECS, Orchestration systems such as Kubernetes, and more…



jahia

Why Docker with Jahia ?

- Useful for development, testing and pre-production
- Easily launch different versions of Jahia
- Simplifies packaging & evaluation of Jahia
- Simplifies upgrade management
- Using Docker Compose you can setup full stacks including database, ElasticSearch, etc…

jahia

jahia

Jahia Docker Images

jahia

Overview

Modern docker images are available starting with Jahia 8.0.3.0

Jahia Docker images:

- are built directly from the codebase
- support all of Jahia features
- follow best practices for Docker containers
- simplify migrations (does not require fix appliers)

Two docker Images

jahia/jahia-ee

- Available at: https://hub.docker.com/r/jahia/jahia-ee/tags
- Production Enterprise image
- Supports “all” Jahia features (incl. all databases)
- Uses release tags (8.1.1.0, ...) & aliases (8, 8.1, 8.1.1, …)

jahia/jahia-discovery

- Available at: https://hub.docker.com/r/jahia/jahia-discovery/tags
- Discovery image
- Pre-installed with Digitall sample site and Derby database
- Uses release tags (8.1.1.0, ...) & aliases (8, 8.1, 8.1.1, …)

jahia

Start Jahia with Docker

Get Started

```
docker run -p 8080:8080 jahia/jahia-ee:8.1.1.2
```

- This is all needed to get started with Jahia on Docker
- It automatically starts Jahia with Derby and a 30 days discovery license

Progressively build your environment

- Connect Jahia to a database
- Provide a license
- Add environment variables
- Start a provisioning script
- And more…

```
docker run -p 8080:8080 \
  -e DB_VENDOR="mariadb" -e DB_HOST="mariadb" \
  -e DB_NAME="jahia" -e DB_USER="jahia" -e DB_PASS="fakepassword" \
  -e JPDA=true \
  -e OPERATING_MODE=development \
  -e JAHIA_PROPERTIES="prop1=xx,prop2=yy" \
  jahia/jahia-ee:8.0.3.0
```

jahia

Container runtimes

There are multiple ways to start a Docker container:

- Using the Docker Engine with the Docker CLI (docker run …)
- Using the Docker Engine with Docker compose
- Using Kubernetes with containerd
- Using cloud native container services such as AWS ECS
- and more…

The Docker Engine is the most common container engine (especially useful for local execution), but other engine do exist

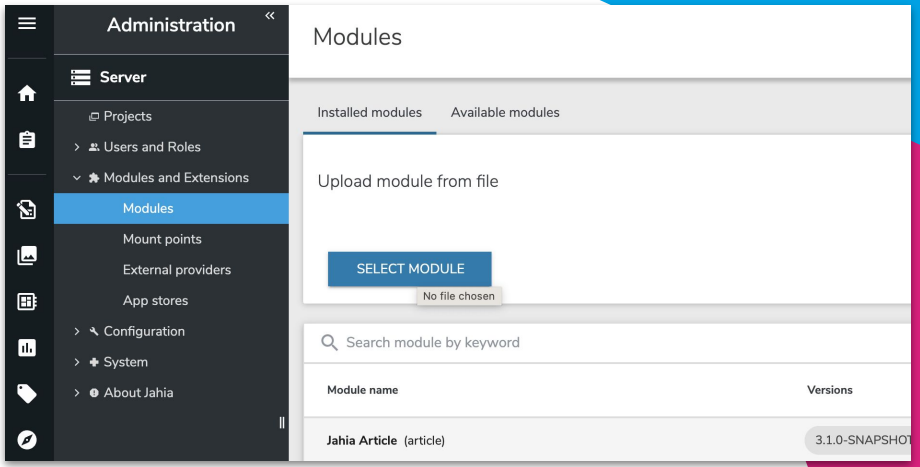
see: https://kubernetes.io/docs/setup/production-environment/container-runtimes/

“docker run” is used primarily for development, other container services (such as Kubernetes) are used for production applications.

jahia

Deploying modules to Jahia Docker containers

- Using the module UI
  - Upload the JAR through the administration UI



1. Using the Jahia Maven plugin

```
mvn jahia:deploy -Djahia.deploy.targetContainerName=DOCKER_CONTAINER_ID
```

(Note: requires Jahia Maven Plugin version >= 5.13)

```
mvn org.jahia.server:jahia-maven-plugin:5.13:deploy
-Djahia.deploy.targetContainerName=DOCKER_CONTAINER_ID
```

2. Using the Provisioning API (later in this presentation)

Debugging Jahia Docker containers

Environment variable : JPDA=true
Startup commands:

```
docker run -e JPDA=true -p 8080:8080 -p 8000:8000 jahia/jahia-ee:latest
```

Note: the port 8000 has to be exposed to be able to connect the Debugger

Connect an IDE to debug using a remote JVM connection to localhost:8000

jahia

Create your own image ?

- Chaining images is part or Docker core principles
  - Using Docker “FROM” statement
  - For example:
    - jahia/jahia-ee is built “from” tomcat
    - jahia/jahia-discovery is built “from” jahia-ee

- Creating your own image using jahia/jahia-ee is supported option:
  - To package additional dependencies
  - To add additional logic specific to your environment
  - To modify the setup of the source image

- Support
  - jahia-ee is the only docker image fully supported. Custom images will be supported to the extent that the issue is reproducible on the jahia-ee images.

More on
https://academy.jahia.com/documentation/system-administrator/dev-ops/docker/building-or-extending-jahia-images

jahia

Upgrading using Docker

- Software upgrade is performed by updating the image associated with a container

- A three steps process
  - Stop the Jahia container(s)
  - Start a new Jahia container in version n+1 that points to the same volume and database
  - If running a cluster, once the first container has finished starting, all other containers can be started

```
docker run [docker options here] --name jahia8 -v [LOCAL_PATH]:/var/jahia
jahia/jahia-ee:8.0.3.0
docker stop jahia8 && docker rm $_
docker run [docker options here] --name jahia8 -v [LOCAL_PATH]:/var/jahia
jahia/jahia-ee:8.1.0.0
```

More on
https://academy.jahia.com/documentation/system-administrator/dev-ops/docker/migrations-with-docker-images

jahia

jahia

Docker Compose

jahia

What is it ?

- Compose is a tool for defining and running multi-container Docker applications.
- YAML file to configure your application’s services (ie containers)
- With a single command, you create and start all the services from your configuration.
- Dependencies can be defined between services

jahia

docker-compose.yml

```
version: '3.6'
services:
  mariadb:
    image: library/mariadb:10-focal
    command: --max_allowed_packet=134217728 --transaction-isolation=READ-UNCOMMITTED --innodb-lock-wait-timeout=10
    networks:
      stack:
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      …

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.5
    networks:
      stack:
    environment:
      - cluster.name=jahia-es-cluster
      …
    ports:
      - 9200:9200
```

jahia

docker-compose.yml

```
kibana:
  image: docker.elastic.co/kibana/kibana:7.17.5
  environment:
    - ELASTICSEARCH_URL=http://elasticsearch:9200
    …
  ports:
    - '8601:5601'
  networks:
    stack:
  depends_on:
    - elasticsearch

jahia:
  image: jahia/jahia-ee-dev:8-SNAPSHOT
  depends_on:
    - mariadb
  ports:
    - 8080:8080
    - 443:8443
  networks:
    stack:
  environment:
    DBMS_TYPE: mariadb
    …
    OPERATING_MODE: development
    PROCESSING_SERVER: 'true'
    …
```

jahia

docker-compose.yml

```
  jcustomer:
    image: jahia/jcustomer-dev:2.1.0-SNAPSHOT
    networks:
      stack:
    environment:
      - UNOMI_ELASTICSEARCH_ADDRESSES=elasticsearch:9200
      …
    ports:
      - 9443:9443
      - 8181:8181
      - 8102:8102
    depends_on:
      - elasticsearch

networks:
  stack:
    driver: bridge
```

jahia

jahia

Jahia Provisioning API

jahia

Overview

An API handling all the operation necessary for a Jahia node to reach a desired state autonomously (no manual operations)

Allow Jahia to supports “infrastructure as code” principles, using YAML or JSON provisioning files describing actions to be performed on a Jahia server

A provisioning scripts contains commands executed sequentially

Useful to put a new Docker image into a desired state, for example before launching automated tests

jahia

How to use it?

Create a provisioning file, for example:

- Install articles v3.0.0 and bookmarks 3.0.0

```
- installOrUpgradeBundle:
  - "mvn:org.jahia.modules/article/3.0.0"
  - "mvn:org.jahia.modules/bookmarks/3.0.0"
  autoStart: true
```

Provide it to your Jahia server with one of the following techniques:

- Using the REST API /modules/api/provisioning
- By placing it on the filesystem “/var/jahia/patches/provisioning/”
  - Can be done using volume bind, docker cp, …
- Using “EXECUTE_PROVISIONING_SCRIPT” environment variable

jahia

Most common commands

- addMavenRepository
  - declares a maven repository
- include
  - executes another provisioning file
- installBundle
  - downloads and installs a module
- enable:
  - enables a module on a site
- editConfiguration
  - edit/create a configuration file
- import
  - imports a zip file previously exported
- executeScript
  - execute a script (groovy, GraphQL)

32 lines (27 sloc) | 1.33 KB — Raw | Blame

```
# For more details about the provisioning API, you can refer to the Academy: https://academy.jahia.com/home
# You can also find some details here: https://github.com/Jahia/jahia-private/blob/master/bundles/provisioning/README.md

# Start with the provisioning script from Tutorial #3
- include: https://raw.githubusercontent.com/Jahia/provisioning-tutorials/main/03-augmented-search/provisioning.yaml

# Install and start bundles for Forms and jExperience
- installBundle:
  - 'mvn:org.jahia.modules/forms-core/3.4.0'
  - 'mvn:org.jahia.modules/forms-snippets-extension/3.1.0'
  - 'mvn:org.jahia.modules/forms-nocss-theme/2.0.0'
  - 'mvn:org.jahia.modules/forms-prefill/2.0.0'
  - 'mvn:org.jahia.modules/forms-extended-inputs/2.0.0'
  - 'mvn:org.jahia.modules/jexperience/2.3.0'
  autoStart: true
  uninstallPreviousVersion: true

# Enable jExperience on digitall
- enable: "jexperience"
  site: "digitall"

# Configure jExperience
- editConfiguration: "org.jahia.modules.jexperience.settings"
  configIdentifier: "global"
  properties:
    jexperience.jCustomerURL: "https://jcustomer:9443"
    jexperience.jCustomerUsername: "karaf"
    jexperience.jCustomerPassword: "karaf"
    jexperience.jCustomerTrustAllCertificates: "true"
    jexperience.jCustomerUsePublicAddressesForAdmin: "false"
    jexperience.jCustomerKey: "670c26d1cc413346c3b2fd9ce65dab41"

```

More on
https://academy.jahia.com/home/documentation/system-administrator/devops/provisioning/provisioning-commands.html

Step by step provisioning

When using the API, Jahia provisioning can be broken down in multiple scripts executed in a particular order.

This provides with fine-grain control over the server provisioning

For example:

- Install a set of commonly available modules
  - The script triggers the installation of “module A”
- Install a single module (“module B”) that was just compiled
  - The new module has a dependency on “module A”
- Perform a configuration on that new module
  - This operation requires the presence of both “module A” and “module B”

Breaking down the provisioning in multiple different scripts (or steps) is the only way to support the scenario detailed above

jahia

jahia

How do we use it?

jahia

Overview

We extensively use Docker and Jahia provisioning API internally at Jahia

It is used:

- Manually
- By our CI environment
- By our preview environment

jahia

Manually

Starting a particular Jahia version is as simple as:

- docker run -p 8080:8080 jahia/jahia-ee:VERSION

It is used across the company (QAs, Devs, Presales, …) to quickly spin-up a Jahia instance

VERSION can be any version that is listed in the Docker Hub images or an alias such as 8, 8.1 or “latest”

Jahia CI environment

Our Continuous Integration environment relies on Docker and Jahia provisioning API to test recent changes to our modules

In particular:

- When a PR is created (successful tests is required for merge)
- After a PR has been merged
- Nightly or Weekly test execution

You can see it in action for one of our modules (graphql-dxm-provider) here:
https://github.com/Jahia/graphql-core/tree/master/tests

jahia

CI environment: How ?

Our workflow:

- The CI platform triggers the creation of a runner
- A Docker image of the tests (Cypress or Selenium) is created
- A startup script is executed:
  - It uses docker-compose to start containers:
    - One or more Jahia and jCustomer containers
    - The test container that was just built
    - Any other containers needed for the tests
- Once containers are started, the test container:
  - Wait for Jahia to be available
  - Submit a provisioning script
  - (optionally) install a recently built module
  - Execute the tests
- The CI collects the results and provides a report

https://github.com/Jahia/graphql-core/tree/master/tests

jahia

Jahia preview environment

Once a day, a new environment is created from scratch, it automatically deploys the latest development snapshot of Jahia and a large set modules

This allows our product managers to quickly access a “complete” environment containing the latest changes

It is sometimes used jointly with our QA to easily demonstrate an issue

No data is persisted from day to day, the environment is always “fresh” (by design)

Preview environment: How ?

Workflow:

- At the beginning of the day, CircleCI triggers creation of an environment via Terraform:
  - One single EC2 Instance of size: t3a.xlarge
  - Terraform runs a startup script (bash) on this new EC2 instance
- The startup script:
  - Uses docker-compose to start the following containers: MariaDB, Elasticsearch (single-node), Kibana, Jahia (snapshot), jCustomer (snapshot)
  - Waits for Jahia and jCustomer to be started
  - Send a provisioning script via the API
  - Execute a set of additional provisioning scripts (Groovy & GraphQL)
- At the end of the day, CircleCI triggers the destruction of the entire environment via terraform

jahia

jahia

Tips & Tricks

jahia

Wait for Jahia

Waiting for Jahia to start might be necessary to prevent the call to the provisioning API from being made before the provisioning bundle has started

```
START_TIME_JAHIA=$SECONDS
echo "$(date +'%d %B %Y - %k:%M') == Waiting for Jahia to startup - check URL ${JAHIA_URL}/cms/login"
while [[ "$(curl -s -o /dev/null -w ''%{http_code}'' ${JAHIA_URL}/cms/login)" != "200" ]];
  do
    echo " == wait for 2 seconds for a response to: ${JAHIA_URL}/cms/login";
    sleep 2;
done
ELAPSED_TIME=$(($SECONDS - START_TIME_JAHIA))
echo "$(date +'%d %B %Y - %k:%M') == Jahia became alive in ${ELAPSED_TIME} seconds"
```

https://github.com/Jahia/graphql-core/blob/master/tests/env.run.sh

jahia

Submit a provisioning script

Using the REST API, you can submit a provisioning file directly

```
echo "$(date +'%d %B %Y - %k:%M') == Executing manifest: ${MANIFEST} =="
curl -u root:${SUPER_USER_PASSWORD} -X POST ${JAHIA_URL}/modules/api/provisioning --form
script="@./run-artifacts/${MANIFEST};type=text/yaml"
```

https://github.com/Jahia/graphql-core/blob/master/tests/env.run.sh

jahia

Install modules

A module can also be sent via the API and installed directly using a specially crafted POST request

You can also submit a set of modules

```
  for file in $(ls -1 *-SNAPSHOT.jar | sort -n)
  do
    echo "$(date +'%d %B %Y - %k:%M') [MODULE_INSTALL] == Submitting module from: $file =="
    curl -u root:${SUPER_USER_PASSWORD} -X POST ${JAHIA_URL}/modules/api/provisioning --form
script='[{"installAndStartBundle":"'"$file"'", "forceUpdate":true}]' --form file=@$file
    echo
    echo "$(date +'%d %B %Y - %k:%M') [MODULE_INSTALL] == Module submitted =="
  done
```

Pay attention to the order in which these modules are submitted, in particular if they are interdependent.

https://github.com/Jahia/graphql-core/blob/master/tests/env.run.sh

jahia

Install scripts

The same apply to scripts, which can be GraphQL or Groovy

```
cd ./scripts || exit 1
for file in $(ls -1 0* | sort -n)
do
  echo "$(date +'%d %B %Y - %k:%M') [SCRIPT] == Submitting script: $file =="
  curl -u root:${SUPER_USER_PASSWORD} -X POST ${JAHIA_URL}/modules/api/provisioning --form
script='[{"executeScript":"'"$file"'"}]' --form file=@$file
  echo "$(date +'%d %B %Y - %k:%M') [SCRIPT] == Script executed =="
done
```

jahia

Demo

jahia

jahia

Learn more

jahia

Provisioning API tutorials

Step by step tutorial to progressively discover Docker and our Provisioning API

https://github.com/Jahia/provisioning-tutorials

jahia

Documentation

Dev-Ops section on the Academy
https://academy.jahia.com/documentation/system-administrator#dev-ops

Jahia Dockerfile and entrypoint script
https://github.com/Jahia/jahia/tree/master/docker/docker-jahia-core

An end-to-end CI setup (graphql-dxm-provider)
https://github.com/Jahia/graphql-core/tree/master/tests

jahia

Q & A

jahia

jahia

Next Webinar January 24th, 2023

Deep Dive into Jahia extensible UI

jahia

Jahia is looking for feedback

Jahia wants to make the developer easier and reduce the learning curve.

Any idea? Wants to give feedback on existing ideas?
shuber@jahia.com, rgauthier@jahia.com

jahia

jahia

Thank you!

jahia