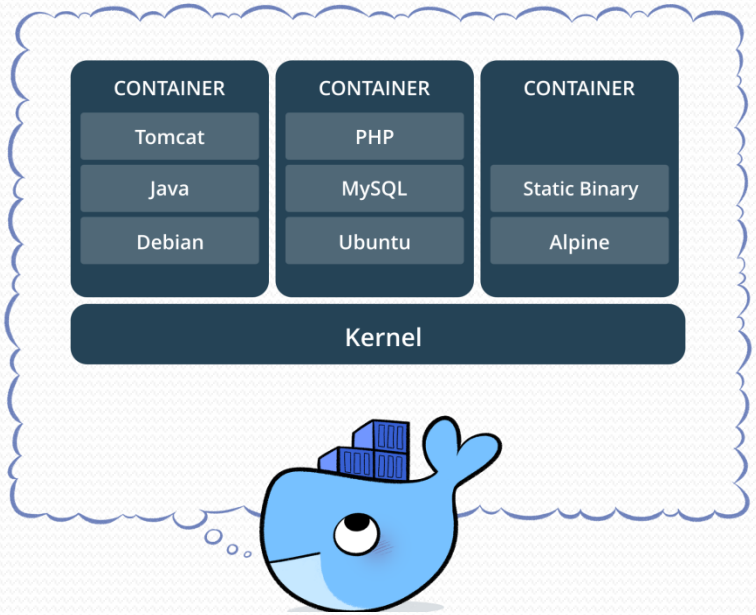
IoT Challenge 2018- Docker platform

16-17 Μαρτίου 2018

Τι είναι το container



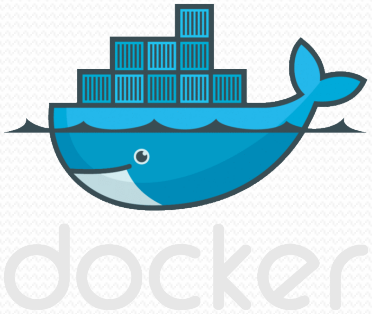
Ένα αυτόνομο εκτελέσιμο κομμάτι λογισμικού το οποίο περιέχει ότι αυτό χρειάζεται για να εκτελεστεί:

- Κώδικα
- Runtime environment
- Εργαλεία συστήματος
- Βιβλιοθήκες
- Ρυθμίσεις

Βασικά οφέλη από τη χρήση docker containers είναι:

- Ταχύτητα, εκτελείται σε λίγα δευτερόλεπτα
- Φορητότητα, μεταφέρεται εύκολα σε άλλη υποδομή
- Αποδοτικότητα, μοιράζεται τους πόρους του συστήματος που το φιλοξενεί
- Επεκτασιμότητα, εύκολη προσθήκη επιπλεόν containers

Docker & components



- Docker
- Docker machine
  - Δημιουργία και διαχείριση απομακρυσμένων υπολογιστών υπο τον έλεγχο του Docker
  - Δημιουργία και διαχείριση Swarm
- Docker compose
  - Εργαλείο αυτοματοποίησης της διαδικασίας δημιουργίας και εκτέλεσης πολλαπλών containers
- Oracle VirtualBox
  - Μηχανή εικονικοποίησης

Containers που χρησιμοποιήθηκαν

- Nodered
  - https://hub.docker.com/r/nodered/node-red-docker/

- WebProtege
  - https://hub.docker.com/r/skyplabs/webprotege/

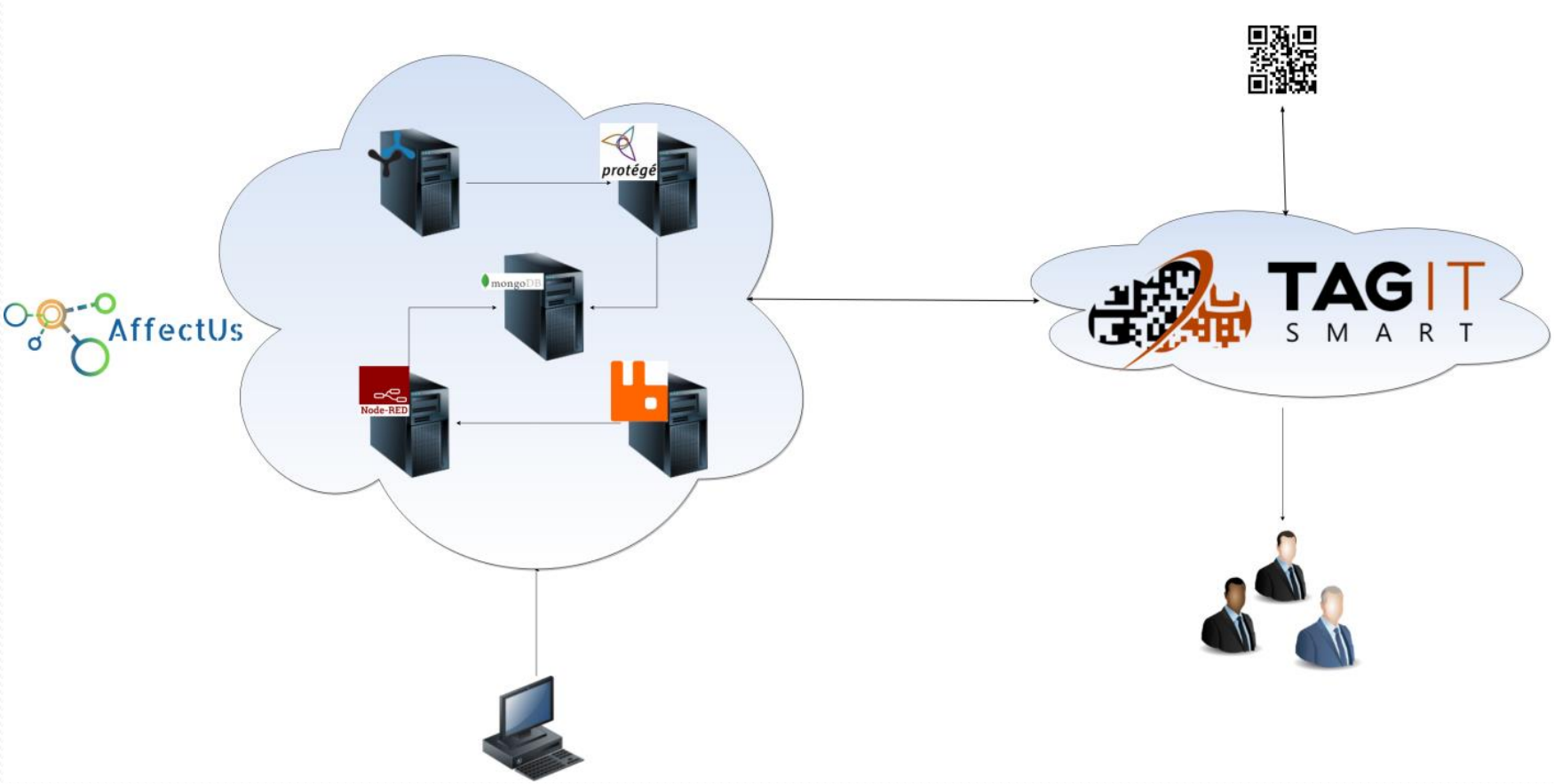
- RabbitMQ + Administrator
  - https://hub.docker.com/_/rabbitmq/

- MongoDB
  - https://hub.docker.com/_/mongo/

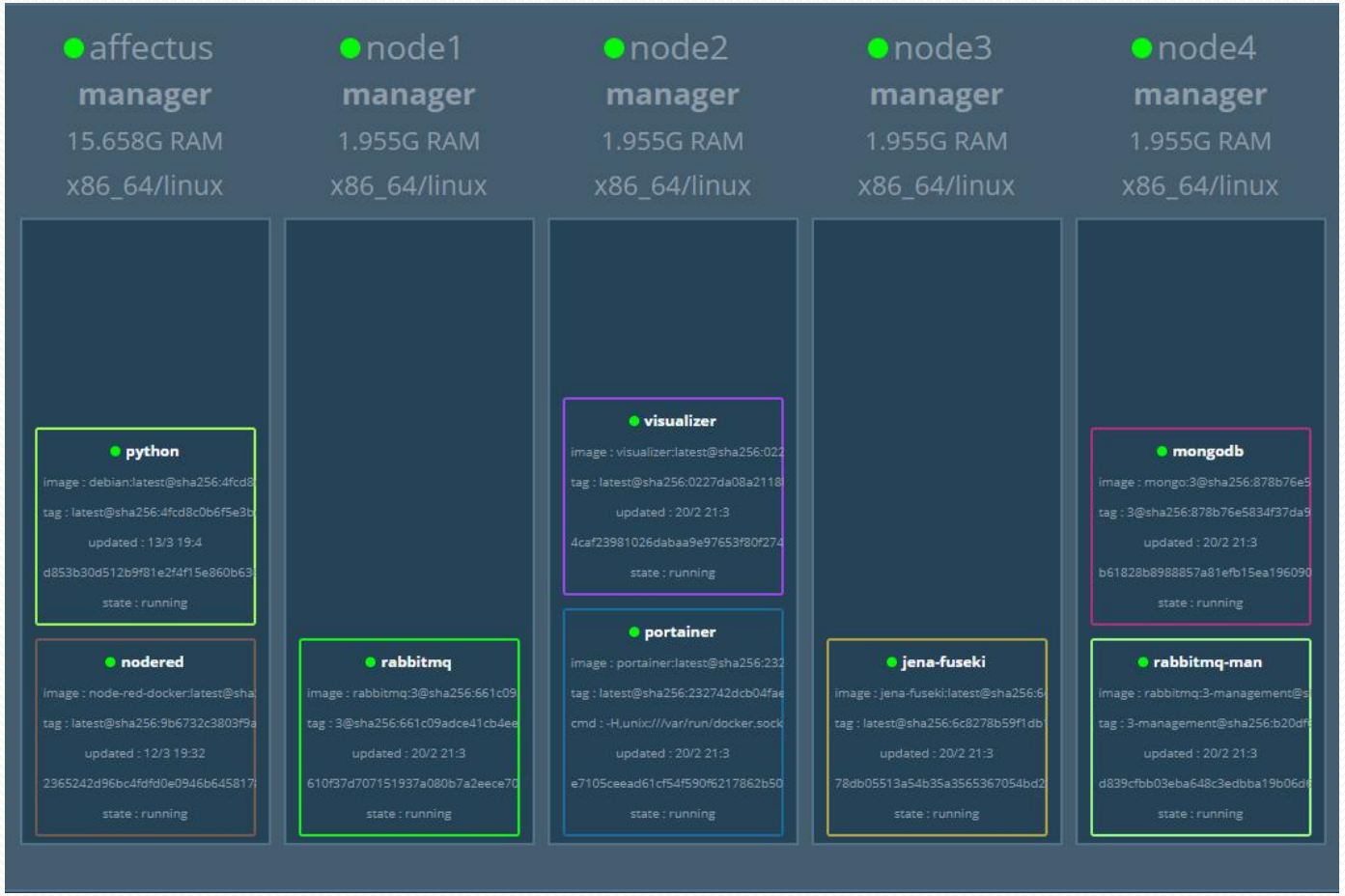
- Jena-Fuseki
  - https://hub.docker.com/r/stain/jena-fuseki/

- Debian (Python)
  - https://hub.docker.com/_/debian/

System Diagram



Swarm deployment



Container setup

- MongoDB
  - Volume: mongodb_data -> /data/db
- WebProtege
  - Volume: webprotege_data -> /data/webprotege
  - Port: 8888 -> 8080
- Jena Fuseki
  - Volume: fuseki_data -> /fuseki
  - Port: 3030 -> 3030
- NodeRed
  - Volume: node-red -> /data
  - Port: 1880 -> 1880
- RabbitMQ
  - Port: 5672 -> 5672
- RabbitMQ Manager
  - Port: 15672 -> 15672
- Debian (Python)
  - Port: 5000 -> 5000

Single step deployment

```
version: "3"
services:

  mongodb:
    image: mongo:3
    volumes:
      - mongodb_data:/data/db
    networks:
      - network_A
      - network_B
    deploy:
      replicas: 1
      update config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure

  webprotege:
    image: docker.io/skyplabs/webprotege
    volumes:
      - webprotege_data:/data/webprotege
    networks:
      - network_A
    ports:
      - 8888:8080
    links:
      - mongodb
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  jena-fuseki:
    image: stain/jena-fuseki
    volumes:
      - fuseki_data:/fuseki
    networks:
      - network_A
    ports:
      - 3030:3030
    environment:
      - ADMIN_PASSWORD
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  nodered:
    image: nodered/node-red-docker
    volumes:
      - node-red:/data
    networks:
      - network_B
    ports:
      - 1880:1880
    links:
      - mongodb
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  rabbitmq:
    image: rabbitmq:3
    networks:
      - network_B
    ports:
      - 5672:5672
    hostname: affectus-rabbitmq
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  rabbitmq-man:
    image: rabbitmq:3-management
    networks:
      - network_B
    ports:
      - 15672:15672
    hostname: affectus-rabbitmq
    links:
      - rabbitmq
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  portainer:
    image: portainer/portainer
    networks:
      - network_A
      - network_B
    ports:
      - 9000:9000
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    deploy:
      placement:
        constraints: [node.role == manager]
      replicas: 1
      restart_policy:
        condition: on-failure

  visualizer:
    image: dockersamples/visualizer
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    deploy:
      placement:
        constraints: [node.role == manager]
    ports:
      - 9090:8080
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  nodered_ssh:
    image: jeroenpeeters/docker-ssh
    networks:
      - network_B
    ports:
      - 2222:22
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      - FILTERS
      - AUTH_MECHANISM=simpleAuth
      - AUTH_USER=root
      - AUTH_PASSWORD
    onrun:
      - npm install sparql-http-client
      - npm install isomorphic-fetch
      - npm install jsonschema
      - npm install shortid
      - npm install sparqljs
      - npm install node-red-contrib-mongodb2
      - npm install node-red-node-mysql
      - npm install node-red-dashboard
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

networks:
  network_A:
  network_B:

volumes:
  mongodb_data:
  webprotege_data:
  fuseki_data:
  node-red:
  /var/run/docker.sock:
```

eval"$(docker-machine env --swarm <name of swarm leader machine>)"**&&** docker-compose –f <name of file>.yml up