Tools

• SCM: [www.github.com](http://www.github.com)
• CI/CD: Jenkins 2.0
  – Important Plugins: Pipeline (for Jenkinsfile), git, github, SSH Slaves (for build slave)
• Platform: docker
• Container Orchestration: docker compose
CI/CD Architecture

- Github
- Jenkinsfile
- Jenkins
- Docker container
- Docker-compose
- Docker-registry
- Dev
- Test
- Stage
- Prod
- Docker engine
- Docker swarm cluster
- Docker swarm cluster
- Docker swarm cluster
- Docker swarm cluster
CI/CD: docker-birthday-3

- docker engine
  - `docker-compose.yml`
  - `Jenkinsfile`
  - `github`
  - `jenkins`
  - `docker container`

- Voting App
- Redis queue
- Java worker
- Results app
- PG database

- volume: `db-data`
docker-birthday-3: voting app

• Birthday App
  – Docker Inc introduce it as training/challenge app
  – A Python webapp which lets you vote between several options
  – A Redis queue which collects new votes
  – A Java worker which consumes votes and stores them in...
  – ...A Postgres database backed by a Docker volume
  – A Node.js webapp showing the results of the voting in real time

Fork: https://github.com/sjeeva/docker-birthday-3
Source: https://github.com/docker/docker-birthday-3
Jenkins 2.0

Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/jenkins_home/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password

Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.

Select plugins to install
Select and install plugins

```json
def node()
    1. stage 'Build'
    2. sh './gradlew assemble'
    3. stage 'Test'
    4. sh './gradlew check'
```

New Item name...

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

External job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

MultiJob Project
MultiJob Project, suitable for running other jobs.

Pipeline
Orchestrate long-running not easily fit in tree-s

Folder
Create a container for multiple things of the

GitHub Organization
Scans a GitHub org

Multibranch Pipeline
Create a set of Pipeline

Build the sudo images for installation
Basic Test: running from repo with preinstalled libs
Basic Test: running from setup.py install
Basic Test: pip develop mode install

Stage View

Average stage times:
(Average full run time: ~27y 22d)

<table>
<thead>
<tr>
<th></th>
<th>Build the sudo images for installation</th>
<th>Basic Test: running from repo with preinstalled libs</th>
<th>Basic Test: running from setup.py install</th>
<th>Basic Test: pip develop mode install</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>1min 50s</td>
<td>1min 40s</td>
<td>1min 29s</td>
<td></td>
</tr>
<tr>
<td>1min 34s</td>
<td>17s</td>
<td>1min 54s</td>
<td>1min 42s</td>
<td>1min 29s</td>
</tr>
<tr>
<td>15s</td>
<td>1min 35s</td>
<td>1min 38s</td>
<td>1min 29s</td>
<td></td>
</tr>
<tr>
<td>16s</td>
<td>1min 40s</td>
<td>1min 42s</td>
<td>1min 29s</td>
<td></td>
</tr>
<tr>
<td>15s</td>
<td>2min 11s</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Containerizing Jenkins...

- Jenkins without SSL
  - Create a docker volume
  - Launch Jenkins by attaching the volume

- Jenkins with SSL
  - Create key and certificate
  - Create volume
  - Launch Jenkins by attaching the volume
  - Launch nginx (apache or haproxy) as a https offloader/reverse proxy for Jenkins

$ docker volume create --name jenkins-data-store
$ docker run -d -p 8888:8080 \
  -v jenkins-data-store:/var/jenkins-home jenkins:2.0

$ cat docker-compose.yml
version: "2"

services:
  jenkins:
    image: jenkins:2.0
    volumes:
      - jenkins-data-store:/var/jenkins_home

  nginx-https:
    build: nginx-https
    ports:
      - 443:443
    depends_on:
      - jenkins
    volumes:
      - jenkins-data-store:

$ cat Dockerfile
FROM nginx
ADD conf.d /etc/nginx/conf.d
ENTRYPOINT ["nginx", ":g", ":daemon off;"]
ls conf.d/
default.conf https-server.crt https-server.key
...Containerizing Jenkins

$ cat conf.d/default.conf
server {
    listen 443 default_server;
    server_name www.example.com;

    ssl on;
    ssl_certificate /etc/nginx/conf.d/https-server.crt;
    ssl_certificate_key /etc/nginx/conf.d/https-server.key;
    ssl_session_cache shared:SSL:10m;

    location / {
        proxy_pass http://jenkins:8080;
        proxy_set_header Host $host;

        # re-write redirects to http as to https, example: /home
        proxy_redirect http:// https://;
    }
}

$ cat genkey.sh
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout conf.d/https-server.key \
    -out conf.d/https-server.crt
Containerized Jenkins & Building Docker Images

- Build Docker image inside a docker container
  - It is not recommended to build on master
  - Jenkins container needs additional privilege
- SSH Node with access to docker engine
  - Create a node
  - Enter “Remote root directory”
    - Used for job workspace
    - must have read, write and access permission for the slave node user
  - Add label ‘docker’
    - required to direct the job to the node
  - Select “Launch slave agents on Unix machines via SSH” and configure “Host” and “Credentials”
Pipeline as a Code

- Traditional
  - UI Based, Manual
  - Create multiple jobs (build, test, deploy)
  - Laborious and not repeatable
- Pipeline as a Code
  - Build, test, and deploy pipeline are implemented in Jenkinsfile
  - Pipeline code can be stored in source control

```bash
$ cat Jenkinsfile
node ('docker') {
  stage "Checkout App Code"
  checkout scm

dir ('example-voting-app') {
  stage "Build Images"
  sh "docker-compose -f docker-compose.yml build"

  stage "Deploy Application"
  sh "docker-compose -f docker-compose.yml -f docker-compose.singlenode.yml stop"
  sh "docker-compose -f docker-compose.yml -f docker-compose.singlenode.yml rm -f"
  sh "docker-compose -f docker-compose.yml -f docker-compose.singlenode.yml up -d"

  stage "Publish Application Details"
  sh "docker-compose -f docker-compose.yml -f docker-compose.singlenode.yml ps"
}
}
Activating the Pipeline

1. Enter an item name: `demo-paac`
   - A job already exists with the name 'demo-paac'

2. Pipeline definition:
   - Definition: Pipeline script from SCM
   - SCM: Git
   - Repository URL: https://github.com/sjeeva/docker-birthday-3.git
   - Credentials: sjeeva/******
   - Branch Specifier: */master
   - Repository browser: (Auto)
   - Additional Behaviours: Add

Freestyle project:
- This is the central feature of Jenkins. Jenkins will build your project, combining any SCM something other than software build.

Pipeline:
- Orchestrates long-running activities that can span multiple build slaves. Suitable for build complex activities that do not easily fit in free-style job type.
Auto-trigger Build

1. sjeeva / docker-birthday-3
   forked from docker/docker-birthday-3
   - Code
   - Pull requests 0
   - Wiki
   - Pulse
   - Graphs
   - Settings

2. Webhooks
   - Webhooks allow external services to be notified when certain events happen within your repository. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

3. Services
   - GitHub integrations directory
     - Jenkins (Git plugin)
     - Jenkins (GitHub plugin)
   - Jenkins hook url
     - Jenkins integration
     - Active
       - We will run this service when an event is triggered.
$ cat docker-compose.yml

version: "2"

services:
  voting-app:
    build: ./voting-app/
    networks:
      - front-tier
      - back-tier

  result-app:
    build: ./result-app/
    networks:
      - front-tier
      - back-tier

  worker:
    image: manomarks/worker
    networks:
      - back-tier

redis:
  image: redis:alpine
  container_name: redis
  ports: ["6379"]
  networks:
    - back-tier

db:
  image: postgres:9.4
  container_name: db
  volumes:
    - "db-data:/var/lib/postgresql/data"
  networks:
    - back-tier

volumes:
  db-data:

networks:
  front-tier:
  back-tier:
$ cat docker-compose.singlenode.yml

version: "2"

services:
  voting-app:
    volumes:
      - ./voting-app:/app
    ports:
      - "5000:80"

  result-app:
    volumes:
      - ./result-app:/app
    ports:
      - "5001:80"