INVENTEURS DU MONDE NUMÉRIQUE

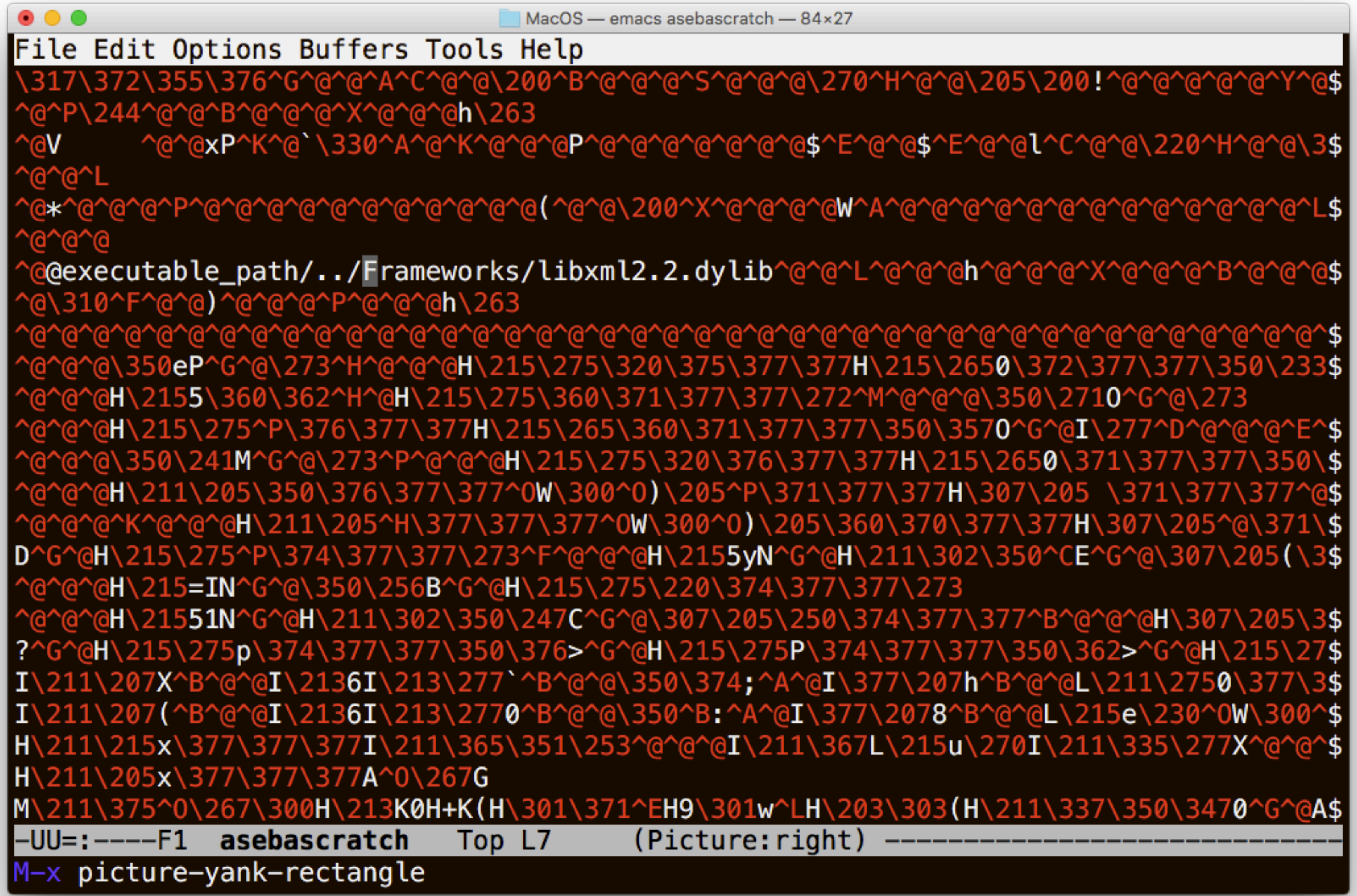
INFRASTRUCTURE AS CODE

David Sherman

EQUIPE
PLEIADE

BORDEAUX SUD-OUEST

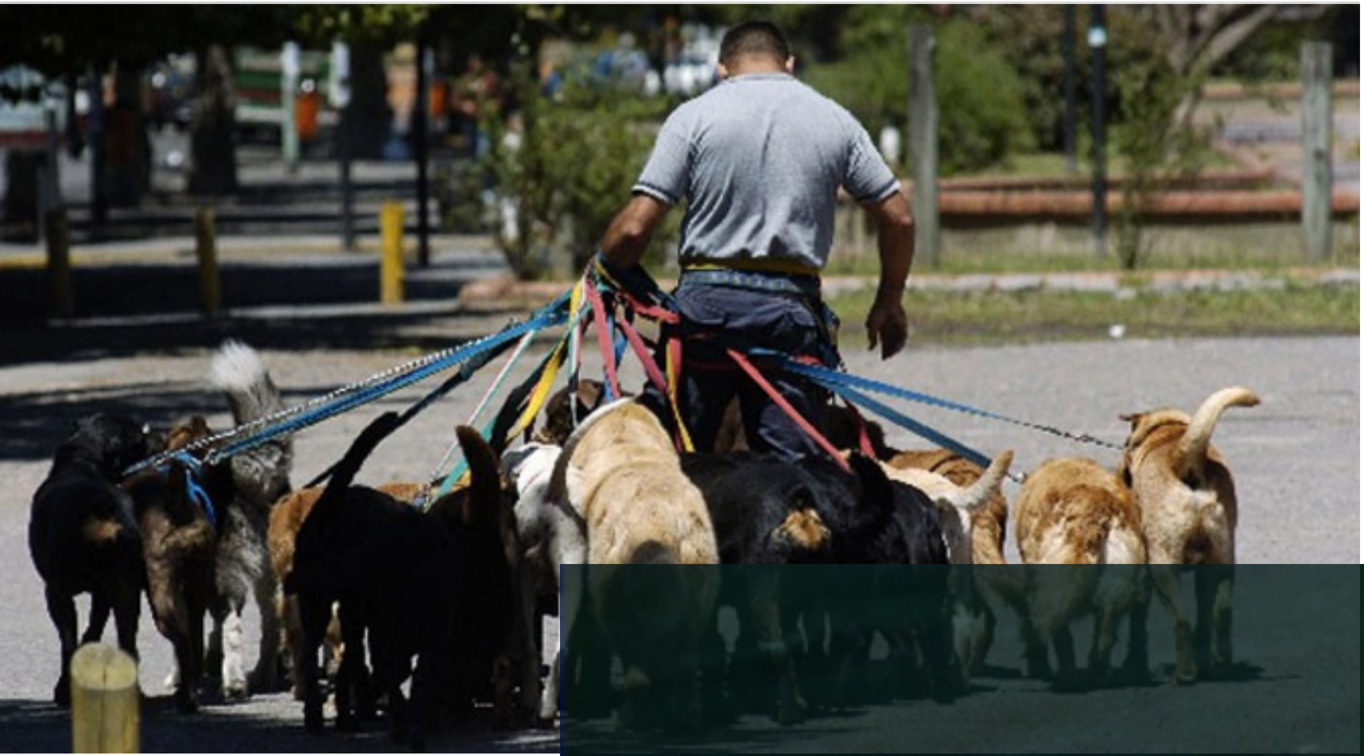
2016-11-08


MacOS — emacs asebascratch — 84×27
File Edit Options Buffers Tools Help

```
\317\372\355\376^G^@^@^A^C^@^@\200^B^@^@^@^S^@^@^@\270^H^@^@\205\200!^@^@^@^@^@^Y^@$
^@^P\244^@^@^B^@^@^@^X^@^@^@h\263
^@V      ^@^@xP^K^@`\330^A^@^K^@^@^@P^@^@^@^@^@^@$^E^@^@$^E^@^@l^C^@^@\220^H^@^@\3$
^@^@^L
^@*^@^@^@^P^@^@^@^@^@^@^@^@^@(^@^@\200^X^@^@^@^@W^A^@^@^@^@^@^@^@^@^@^@^@^@^@^L$
^@^@^@
^@@executable_path/../Frameworks/libxml2.2.dylib^@^@^L^@^@^@h^@^@^@^X^@^@^@^B^@^@^@$
^@\310^F^@^@)^@^@^@^P^@^@^@h\263
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@$
^@^@^@\350eP^G^@\273^H^@^@^@H\215\275\320\375\377\377H\215\2650\372\377\377\350\233$
^@^@^@H\2155\360\362^H^@H\215\275\360\371\377\377\272^M^@^@^@\350\2710^G^@\273
^@^@^@H\215\275^P\376\377\377H\215\265\360\371\377\377\350\3570^G^@I\277^D^@^@^@^E^$
^@^@^@\350\241M^G^@\273^P^@^@^@H\215\275\320\376\377\377H\215\2650\371\377\377\350\$
^@^@^@H\211\205\350\376\377\377^OW\300^O)\205^P\371\377\377H\307\205 \371\377\377^@$
^@^@^@^K^@^@^@H\211\205^H\377\377\377^OW\300^O)\205\360\370\377\377H\307\205^@\371\$
D^G^@H\215\275^P\374\377\377\273^F^@^@^@H\2155yN^G^@H\211\302\350^CE^G^@\307\205(\3$
^@^@^@H\215=IN^G^@\350\256B^G^@H\215\275\220\374\377\377\273
^@^@^@H\21551N^G^@H\211\302\350\247C^G^@\307\205\250\374\377\377^B^@^@^@H\307\205\3$
?^G^@H\215\275p\374\377\377\350\376>^G^@H\215\275P\374\377\377\350\362>^G^@H\215\27$
I\211\207X^B^@^@I\2136I\213\277`^B^@^@\350\374;^A^@I\377\207h^B^@^@L\211\2750\377\3$
I\211\207(^B^@^@I\2136I\213\2770^B^@^@\350^B:^A^@I\377\2078^B^@^@L\215e\230^OW\300^$
H\211\215x\377\377\377I\211\365\351\253^@^@^@I\211\367L\215u\270I\211\335\277X^@^@^$
H\211\205x\377\377\377A^O\267G
M\211\375^O\267\300H\213K0H+K(H\301\371^EH9\301w^LH\203\303(H\211\337\350\3470^G^@A$
-UU=:----F1  asebascratch   Top L7     (Picture:right) ----------------------------
M-x picture-yank-rectangle
```

Pets versus Cattle

Pets versus Cattle



(or sheep)

https://vimeo.com/4486963

Infrastructure as Code

Jenkins 2 Pipeline

- *Jenkinsfile*
- Require environment
- Define stages for build, test, deploy, …
- Run by Jenkins on agents
- Checked in to SCM: versions, branches, dev workflow incl. reviews

Docker

- *Dockerfile*
- Create environment
- Define filesystem layers for individual microservices
- Run on container host
- Checked in to SCM: versions, branches, dev workflow incl. reviews

OUTLINE

1. Jenkins 2 Pipeline

   Syntax, examples

   Blue Ocean user interface

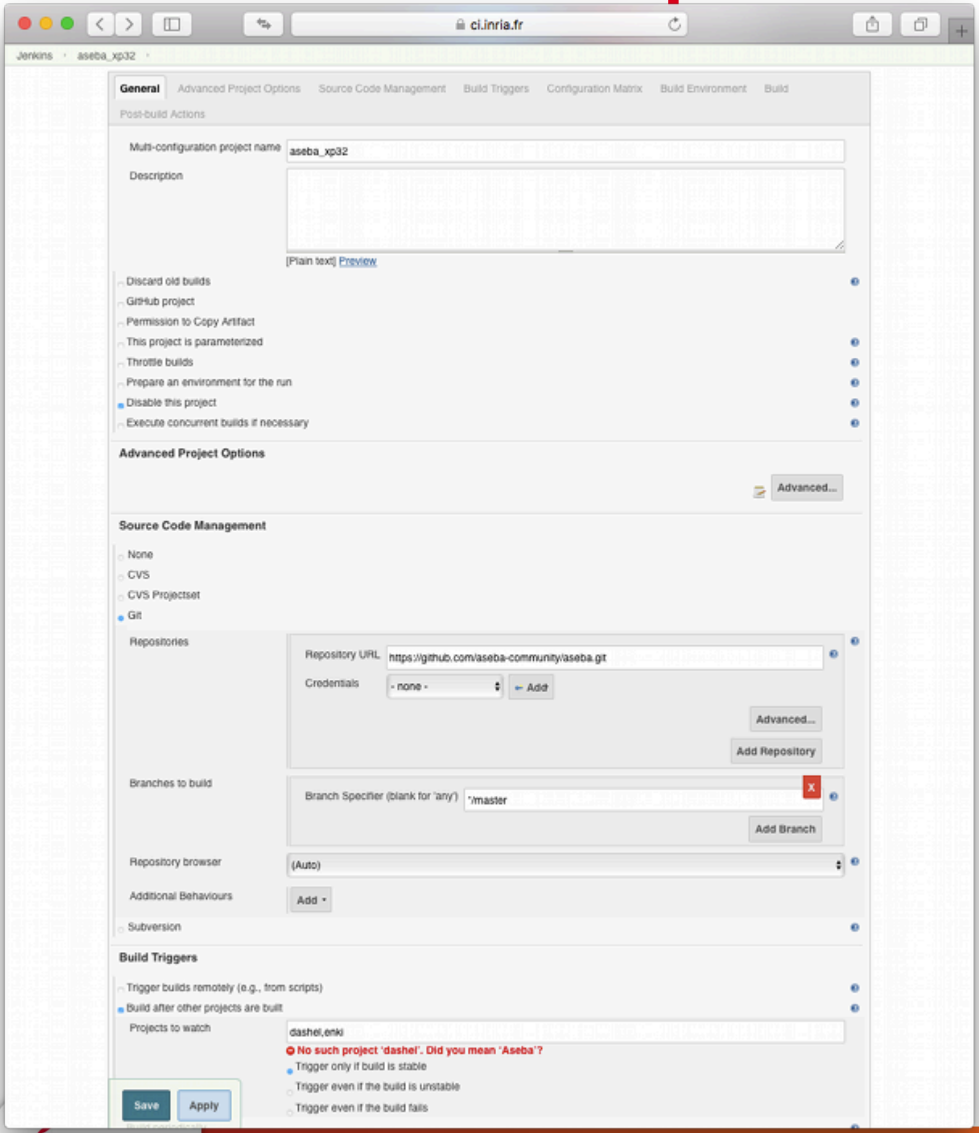
2. Docker

   Syntax, examples

   Orchestration of microservice architectures

3. Take home message: everything in SCM

Jenkins 2 Pipeline: motivation



ci.inria.fr
Jenkins › aseba_xp32 ›
General | Advanced Project Options | Source Code Management | Build Triggers | Configuration Matrix | Build Environment | Build | Post-build Actions

Multi-configuration project name: aseba_xp32

Description:

[Plain text] Preview

- Discard old builds
- GitHub project
- Permission to Copy Artifact
- This project is parameterized
- Throttle builds
- Prepare an environment for the run
- [✓ Disable this project]
- Execute concurrent builds if necessary

Advanced Project Options

Advanced...

Source Code Management

- None
- CVS
- CVS Projectset
- [✓ Git]

Repositories — Repository URL: https://github.com/aseba-community/aseba.git

Credentials: - none - | Add

Advanced... | Add Repository

Branches to build — Branch Specifier (blank for 'any'): */master

Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

- Subversion

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- [✓ Build after other projects are built]

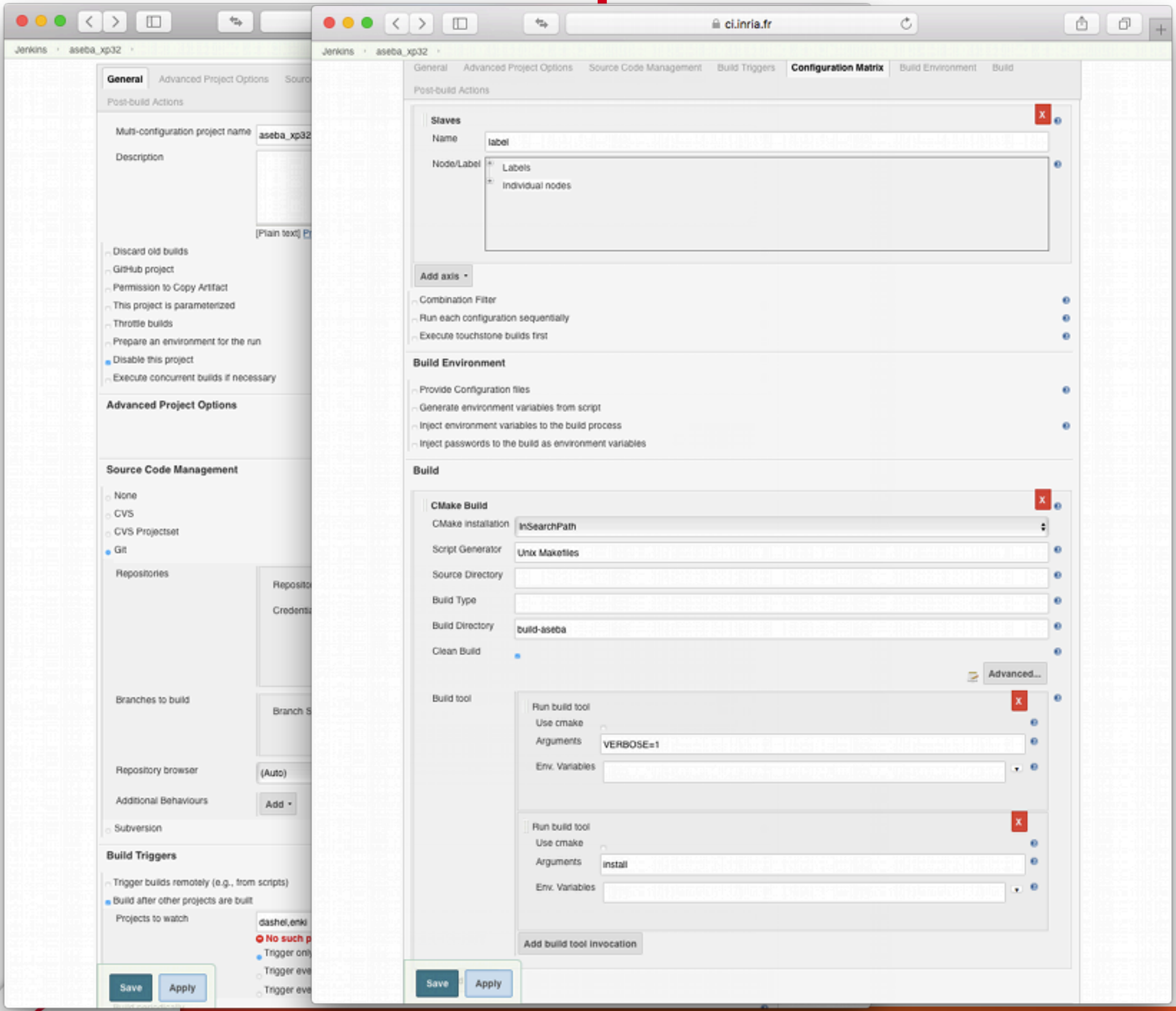
Projects to watch: dashel,enki

No such project 'dashel'. Did you mean 'Aseba'?

- [✓ Trigger only if build is stable]
- Trigger even if the build is unstable
- Trigger even if the build fails

Save | Apply

Jenkins 2 Pipeline: motivation



Jenkins › aseba_xp32 ›

General | Advanced Project Options | Sourc
Post-build Actions

Multi-configuration project name: aseba_xp32

Description

[Plain text] Pr

- Discard old builds
- GitHub project
- Permission to Copy Artifact
- This project is parameterized
- Throttle builds
- Prepare an environment for the run
- Disable this project
- Execute concurrent builds if necessary

Advanced Project Options

Source Code Management

- None
- CVS
- CVS Projectset
- Git

Repositories: Reposito / Credentia

Branches to build: Branch S

Repository browser: (Auto)

Additional Behaviours: Add

- Subversion

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built

Projects to watch: dashel,enki

No such p

- Trigger only
- Trigger eve
- Trigger eve

Save | Apply

ci.inria.fr

Jenkins › aseba_xp32 ›

General | Advanced Project Options | Source Code Management | Build Triggers | **Configuration Matrix** | Build Environment | Build
Post-build Actions

Slaves

Name: label

Node/Label:
- Labels
- Individual nodes

Add axis

- Combination Filter
- Run each configuration sequentially
- Execute touchstone builds first

Build Environment

- Provide Configuration files
- Generate environment variables from script
- Inject environment variables to the build process
- Inject passwords to the build as environment variables

Build

CMake Build

CMake installation: InSearchPath

Script Generator: Unix Makefiles

Source Directory:

Build Type:

Build Directory: build-aseba

Clean Build

Advanced...

Build tool:

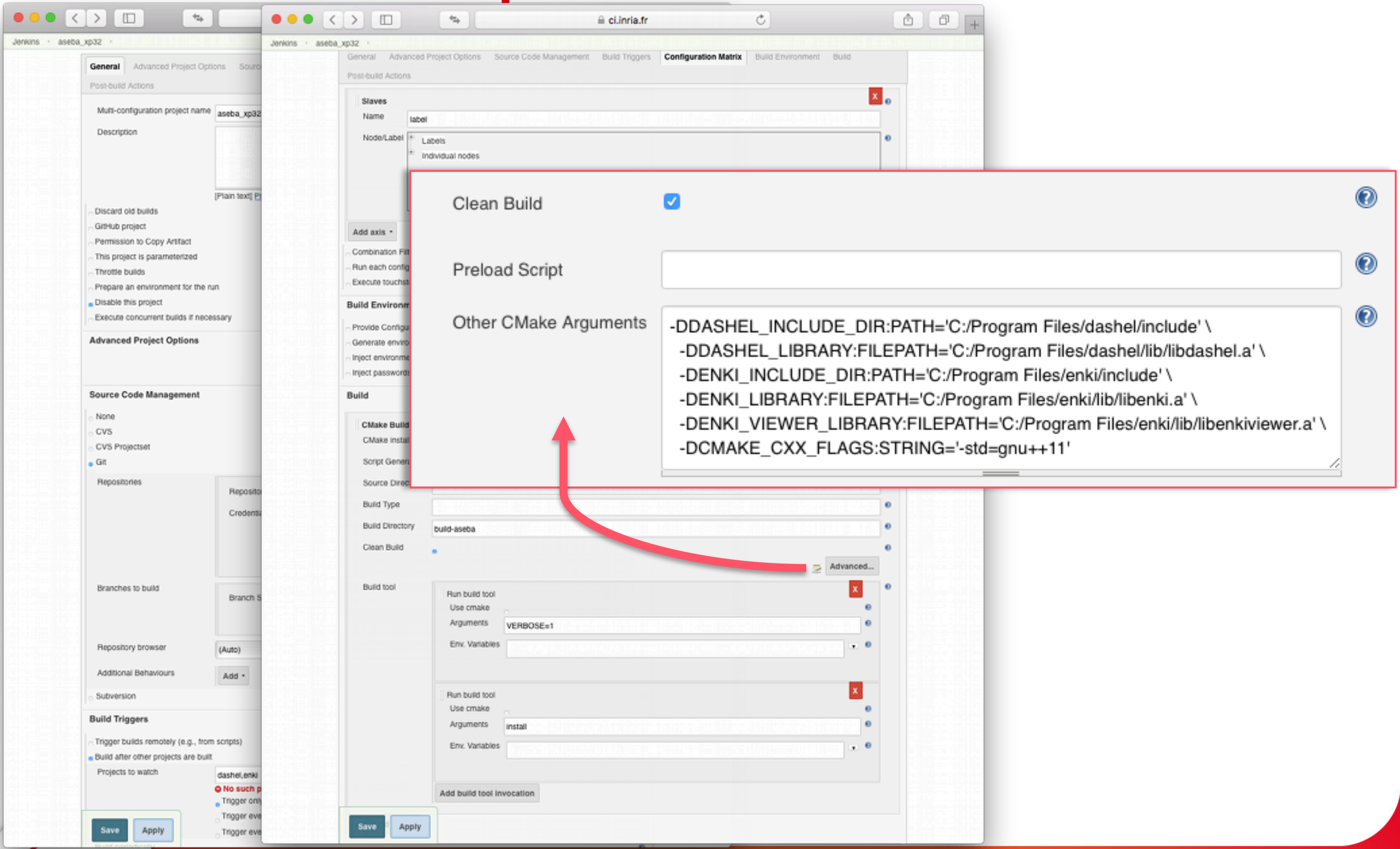
Run build tool
Use cmake
Arguments: VERBOSE=1
Env. Variables:

Run build tool
Use cmake
Arguments: install
Env. Variables:

Add build tool invocation

Save | Apply

Jenkins 2 Pipeline: motivation



Clean Build ✓

Preload Script

Other CMake Arguments

```
-DDASHEL_INCLUDE_DIR:PATH='C:/Program Files/dashel/include' \
  -DDASHEL_LIBRARY:FILEPATH='C:/Program Files/dashel/lib/libdashel.a' \
  -DENKI_INCLUDE_DIR:PATH='C:/Program Files/enki/include' \
  -DENKI_LIBRARY:FILEPATH='C:/Program Files/enki/lib/libenki.a' \
  -DENKI_VIEWER_LIBRARY:FILEPATH='C:/Program Files/enki/lib/libenkiviewer.a' \
  -DCMAKE_CXX_FLAGS:STRING='-std=gnu++11'
```

Multi-configuration project name: aseba_xp32

Slaves — Name: label — Node/Label: Labels, Individual nodes

Build Directory: build-aseba

Run build tool — Use cmake — Arguments: VERBOSE=1

Run build tool — Use cmake — Arguments: install

Source Code Management: Git

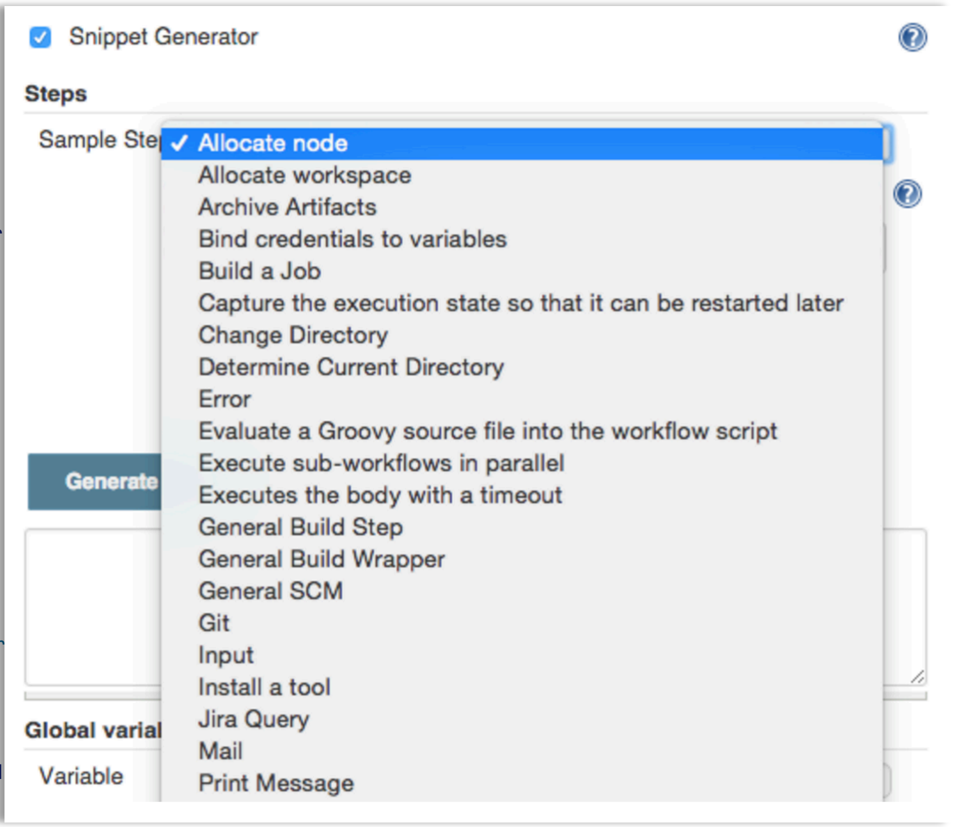
Build after other projects are built — Projects to watch: dashel,enki

Jenkins 2 Pipeline: syntactic structure

```
def servers

stage('Dev') {
    node {
        checkout scm
        servers = load 'servers.groovy'
        mvn '-o clean package'
        dir('target') {stash name: 'war', includes: 'x.war'}
    }
}

stage('QA') {
    parallel(longerTests: {
        runTests(servers, 30)
    }, quickerTests: {
        runTests(servers, 20)
    })
}

milestone 1
stage('Staging') {
    lock(resource: 'staging-server', inversePrecedence: true) {
        milestone 2
        node {
            servers.deploy 'staging'
        }
        input message: "Does ${jettyUrl}staging/ look good?"
    }
    try {
        checkpoint('Before production')
    } catch (NoSuchMethodError _) {
        echo 'Checkpoint feature available in CloudBees Jenkins Enterprise.'
    }
}

milestone 3
stage ('Production') {
```

Jenkins 2 Pipeline: syntactic structure

```
def servers

stage('Dev') {
    node {
        checkout scm
        servers = load 'servers.groovy'
        mvn '-o clean package'
        dir('target') {stash name: 'war', includes: 'x.war
    }
}

stage('QA') {
    parallel(longerTests: {
        runTests(servers, 30)
    }, quickerTests: {
        runTests(servers, 20)
    })
}

milestone 1
stage('Staging') {
    lock(resource: 'staging-server', inversePrecedence: tr
        milestone 2
        node {
            servers.deploy 'staging'
        }
        input message: "Does ${jettyUrl}staging/ look good
    }
    try {
        checkpoint('Before production')
    } catch (NoSuchMethodError _) {
        echo 'Checkpoint feature available in CloudBees Jenkins Enterprise.'
    }
}

milestone 3
stage ('Production') {
```


Snippet Generator

Steps

Sample Ste

✓ Allocate node
Allocate workspace
Archive Artifacts
Bind credentials to variables
Build a Job
Capture the execution state so that it can be restarted later
Change Directory
Determine Current Directory
Error
Evaluate a Groovy source file into the workflow script
Execute sub-workflows in parallel
Executes the body with a timeout
General Build Step
General Build Wrapper
General SCM
Git
Input
Install a tool
Jira Query
Mail
Print Message

Generate

Global varial

Variable

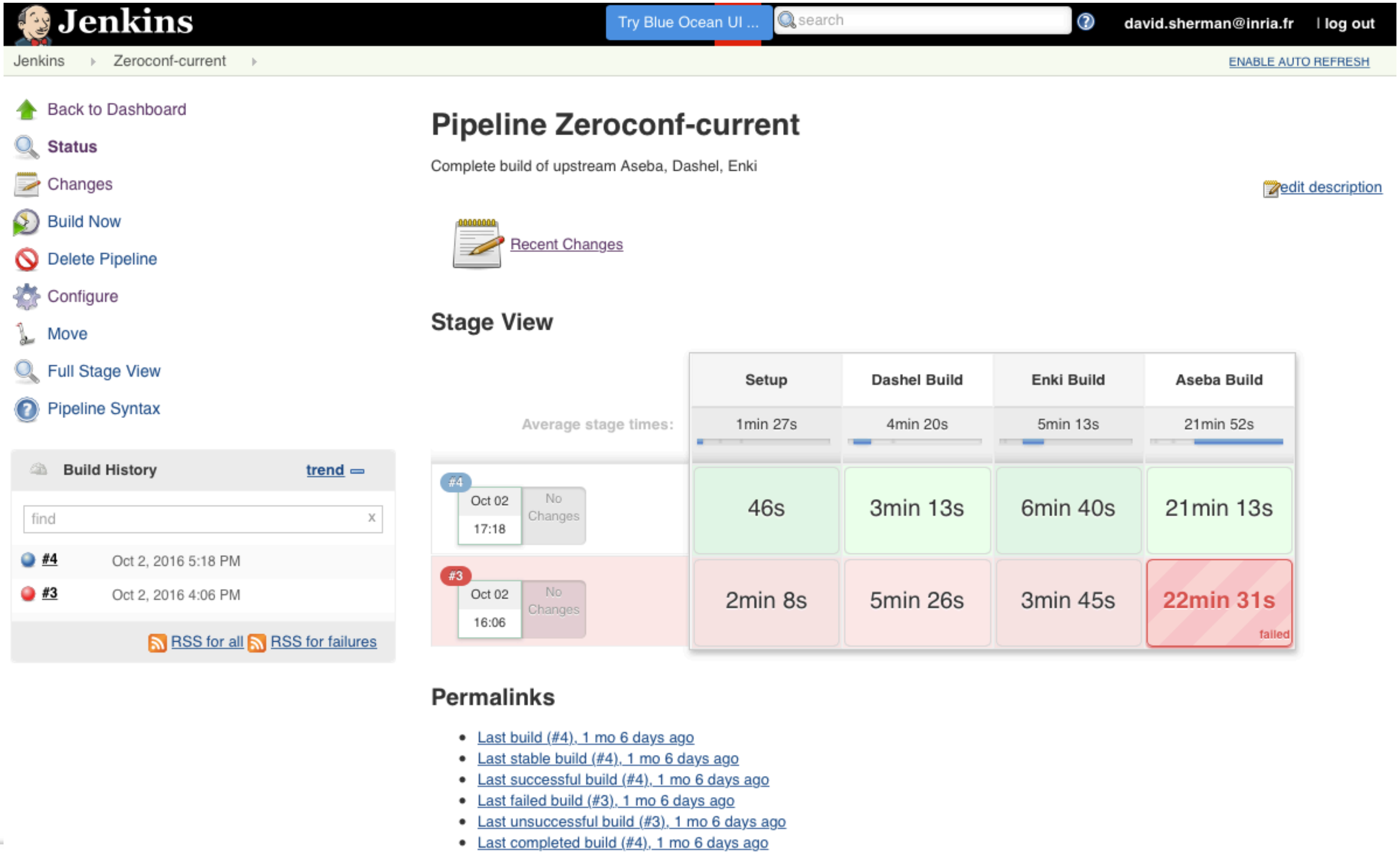
Jenkins 2 Pipeline: model definition

```
pipeline {
     agent any
     stages {
         stage('Build') {
             steps {
                 sh 'echo building...'
             }
         }
         stage('Test') {
             steps {
                 sh 'echo testing...'
             }
         }
         stage('Sanity check') {
             steps {
                 input "Does the staging environment for ${env.APP_NAME} look ok?”
             }
         }
         stage('Deploy - Staging') {
             steps {
                 sh 'echo deploying to staging...'
                 sh 'echo smoke tests...'
             }
         }
         stage('Deploy - Production') {
             steps {
                 sh 'echo deploying to production...'
             }
         }
```

Jenkins 2 Pipeline: model definition

```
stage('Dashel') {
    node('inirobot-win7') {
        git branch: 'pollsocketstream', url: 'https://github.com/davidjsherman/dashel.git’
        sh '''git submodule update --init'''
        withEnv(["INSTALL=${pwd()}/_install","BUILD=${pwd()}/_build"]) {
            sh '''rm -rf ${INSTALL}/dashel && mkdir -p ${INSTALL}/dashel
                  rm -rf ${BUILD} && mkdir ${BUILD}
                  cd ${BUILD}
                  cmake ${WORKSPACE} -G 'Unix Makefiles' \
                        -DCMAKE_INSTALL_PREFIX:PATH=${INSTALL}/dashel \
                        -DBUILD_SHARED_LIBS:BOOL=OFF
                  make
                  make install
               '''
        }
        stash includes: '_install/dashel/**', name: 'dashel’
    }
}
```

Jenkins 2 Pipeline: example



Jenkins | Try Blue Ocean UI ... | search | david.sherman@inria.fr | log out

Jenkins › Zeroconf-current › ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Build Now
- Delete Pipeline
- Configure
- Move
- Full Stage View
- Pipeline Syntax

Build History trend

find

#4	Oct 2, 2016 5:18 PM
#3	Oct 2, 2016 4:06 PM

RSS for all RSS for failures

Pipeline Zeroconf-current

Complete build of upstream Aseba, Dashel, Enki

edit description

Recent Changes

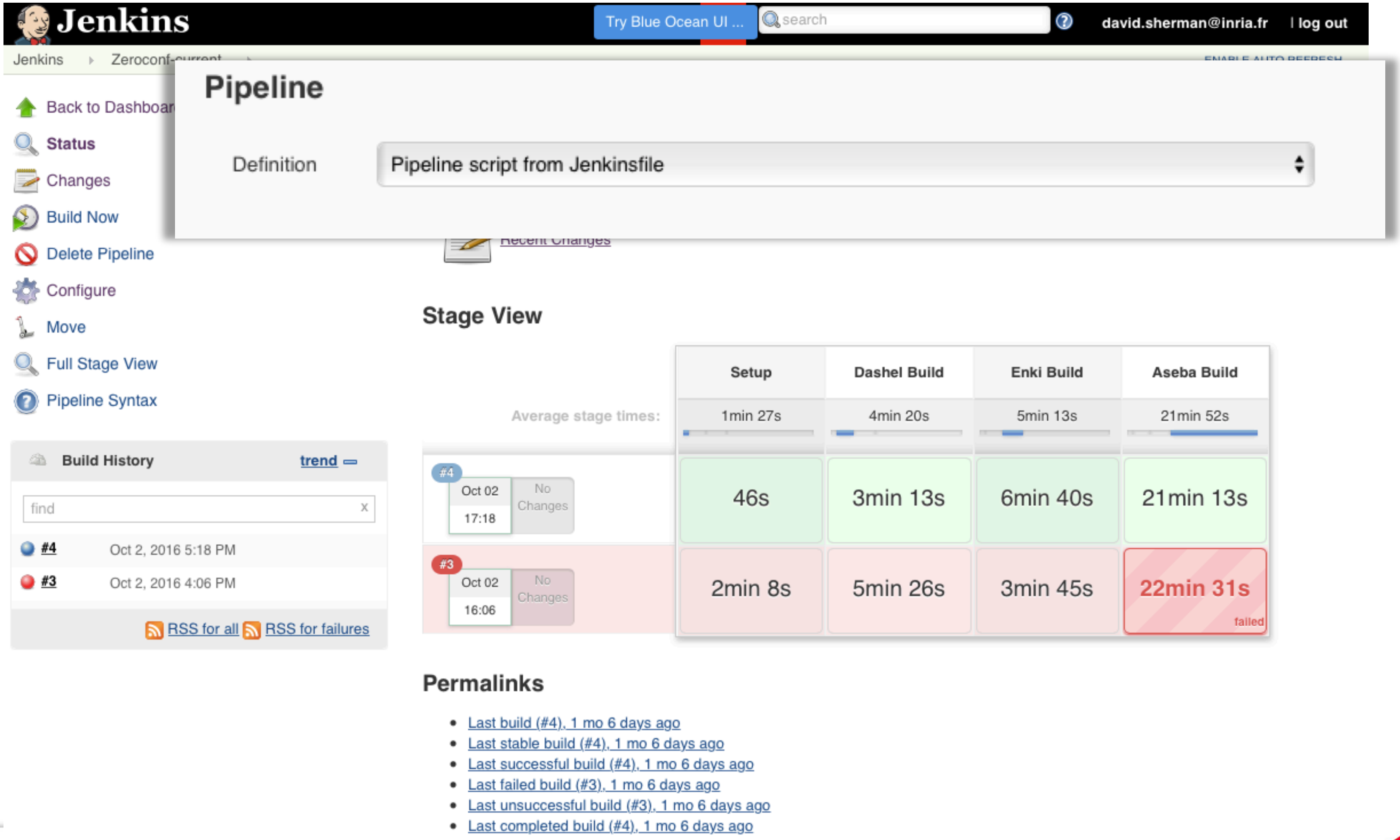
Stage View

	Setup	Dashel Build	Enki Build	Aseba Build
Average stage times:	1min 27s	4min 20s	5min 13s	21min 52s
#4 Oct 02 17:18 No Changes	46s	3min 13s	6min 40s	21min 13s
#3 Oct 02 16:06 No Changes	2min 8s	5min 26s	3min 45s	22min 31s failed

Permalinks

- Last build (#4), 1 mo 6 days ago
- Last stable build (#4), 1 mo 6 days ago
- Last successful build (#4), 1 mo 6 days ago
- Last failed build (#3), 1 mo 6 days ago
- Last unsuccessful build (#3), 1 mo 6 days ago
- Last completed build (#4), 1 mo 6 days ago

Jenkins 2 Pipeline: example



Pipeline

Definition: Pipeline script from Jenkinsfile

Stage View

	Setup	Dashel Build	Enki Build	Aseba Build
Average stage times:	1min 27s	4min 20s	5min 13s	21min 52s
#4 Oct 02 17:18 No Changes	46s	3min 13s	6min 40s	21min 13s
#3 Oct 02 16:06 No Changes	2min 8s	5min 26s	3min 45s	22min 31s failed

Build History — trend

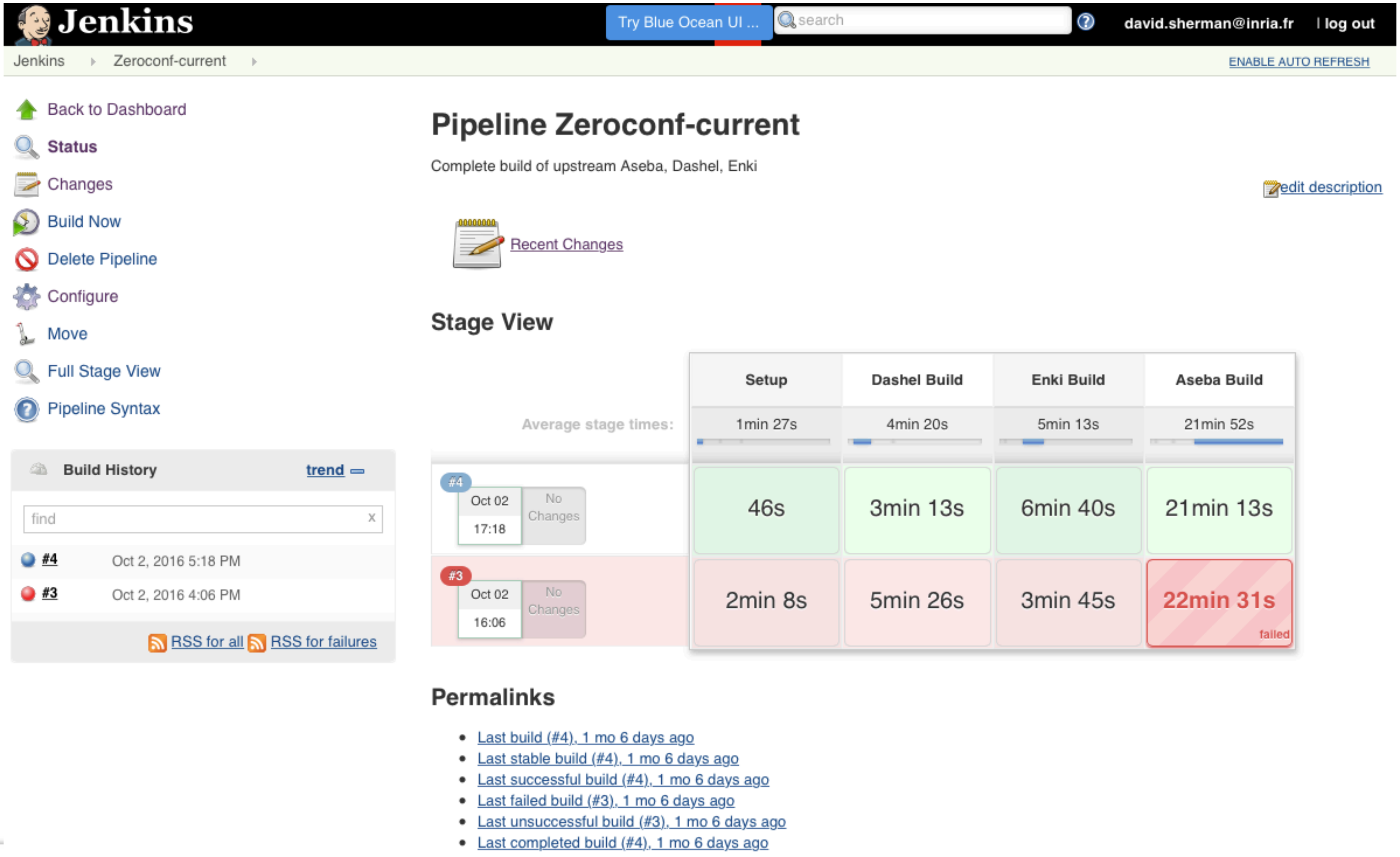
#4	Oct 2, 2016 5:18 PM
#3	Oct 2, 2016 4:06 PM

RSS for all RSS for failures

Permalinks

- Last build (#4), 1 mo 6 days ago
- Last stable build (#4), 1 mo 6 days ago
- Last successful build (#4), 1 mo 6 days ago
- Last failed build (#3), 1 mo 6 days ago
- Last unsuccessful build (#3), 1 mo 6 days ago
- Last completed build (#4), 1 mo 6 days ago

Jenkins 2 Pipeline: example



Jenkins | Try Blue Ocean UI ... | search | david.sherman@inria.fr | log out

Jenkins › Zeroconf-current › ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Build Now
- Delete Pipeline
- Configure
- Move
- Full Stage View
- Pipeline Syntax

Build History trend

find

#4	Oct 2, 2016 5:18 PM
#3	Oct 2, 2016 4:06 PM

RSS for all RSS for failures

Pipeline Zeroconf-current

Complete build of upstream Aseba, Dashel, Enki

edit description

Recent Changes

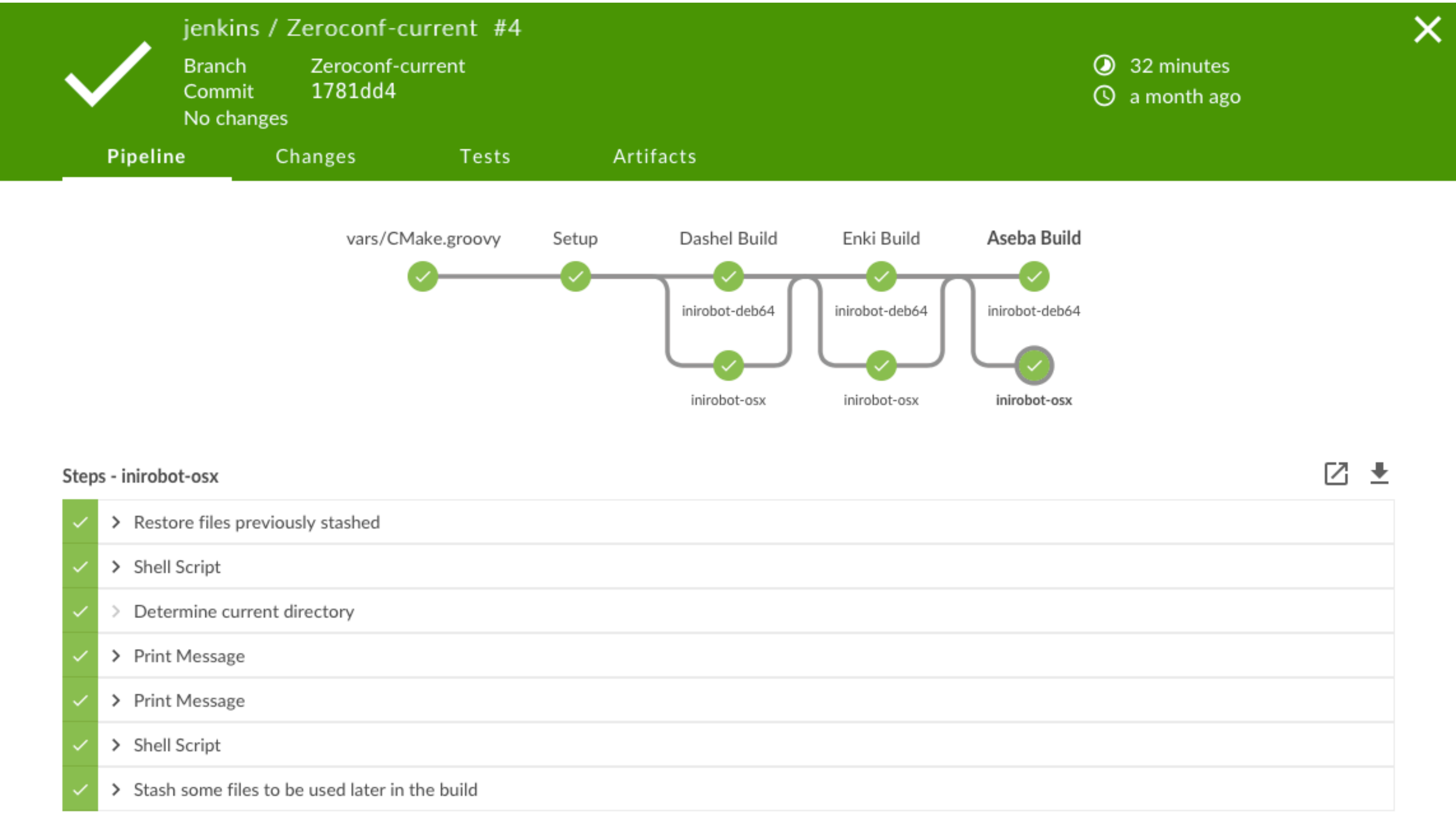
Stage View

	Setup	Dashel Build	Enki Build	Aseba Build
Average stage times:	1min 27s	4min 20s	5min 13s	21min 52s
#4 Oct 02 17:18 No Changes	46s	3min 13s	6min 40s	21min 13s
#3 Oct 02 16:06 No Changes	2min 8s	5min 26s	3min 45s	22min 31s failed

Permalinks

- Last build (#4), 1 mo 6 days ago
- Last stable build (#4), 1 mo 6 days ago
- Last successful build (#4), 1 mo 6 days ago
- Last failed build (#3), 1 mo 6 days ago
- Last unsuccessful build (#3), 1 mo 6 days ago
- Last completed build (#4), 1 mo 6 days ago

Jenkins 2 Pipeline: example


jenkins / Zeroconf-current #4

Branch Zeroconf-current
Commit 1781dd4
No changes

32 minutes
a month ago

Steps - inirobot-osx

- Restore files previously stashed
- Shell Script
- Determine current directory
- Print Message
- Print Message
- Shell Script
- Stash some files to be used later in the build

Jenkins 2 Pipeline: multibranch

For a multibranch project, Jenkins will:

- Check out each of the (selected) branches
- Create a workspace for each branch
- Run the `Jenkinsfile` in each branch

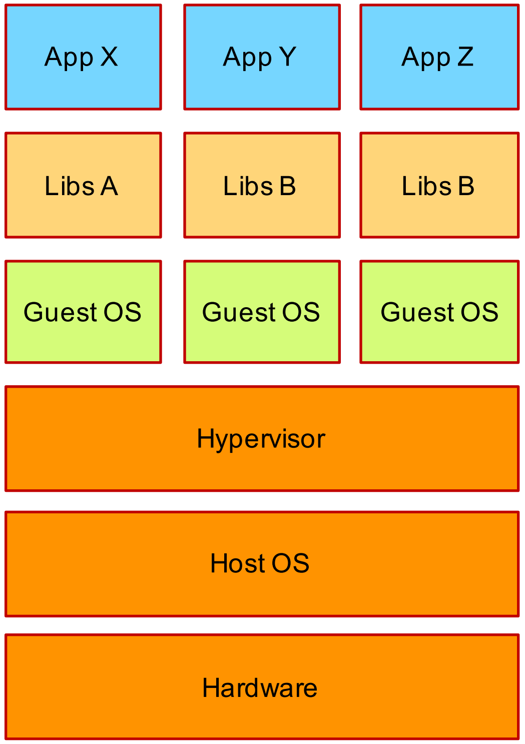
This can be extended to all repositories in a given GitHub organization

This can be used to automatically trigger building and testing the merge for every pull request that is submitted

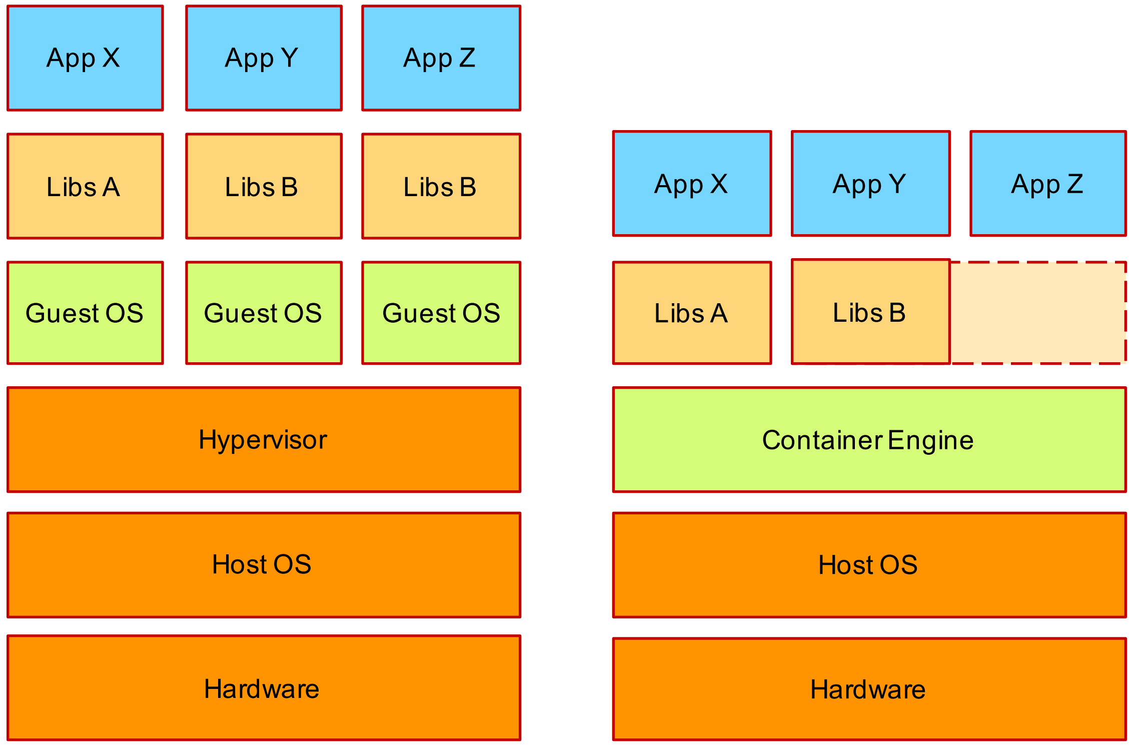
Since the `Jenkinsfile` is shipped with the source code, building and testing

- Are integrated in the development process
- Benefit from the team's **git** workflow (issues, reviews, branch staging)

Docker: motivation

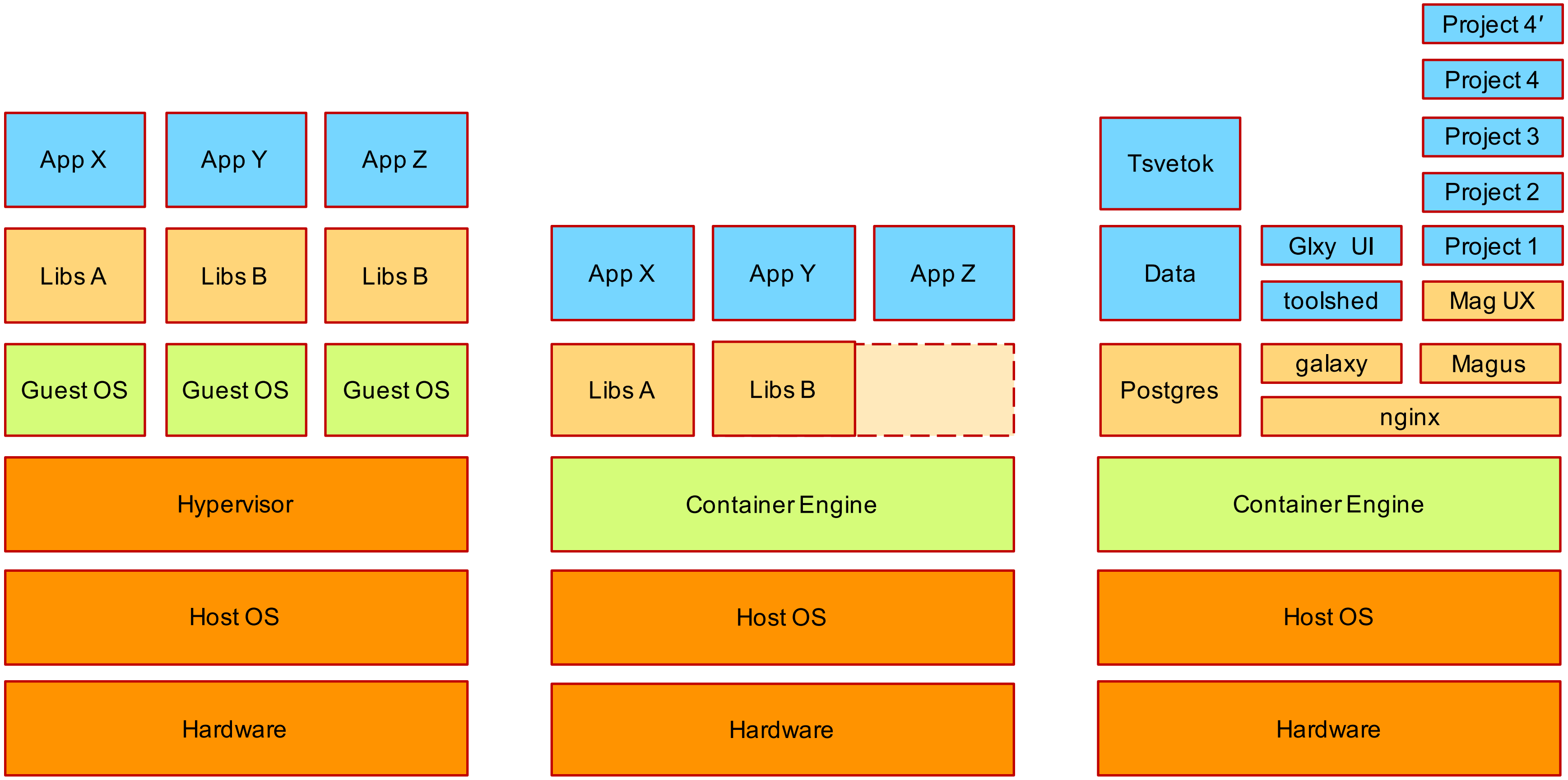


Docker: motivation



A *container* encapsulates one application and its dependencies

Docker: motivation



A *container* encapsulates one application and its dependencies

Docker: syntactic structure

A *container* encapsulates one application and its dependencies

```
FROM        ubuntu:14.04
RUN         apt-get update && apt-get install -y redis-server
EXPOSE      6379
ENTRYPOINT  ["/usr/bin/redis-server"]
```

Docker: syntactic structure

A *container* encapsulates one application and its dependencies

```
FROM        ubuntu:14.04
RUN         apt-get update && apt-get install -y redis-server
EXPOSE      6379
ENTRYPOINT  ["/usr/bin/redis-server"]
```

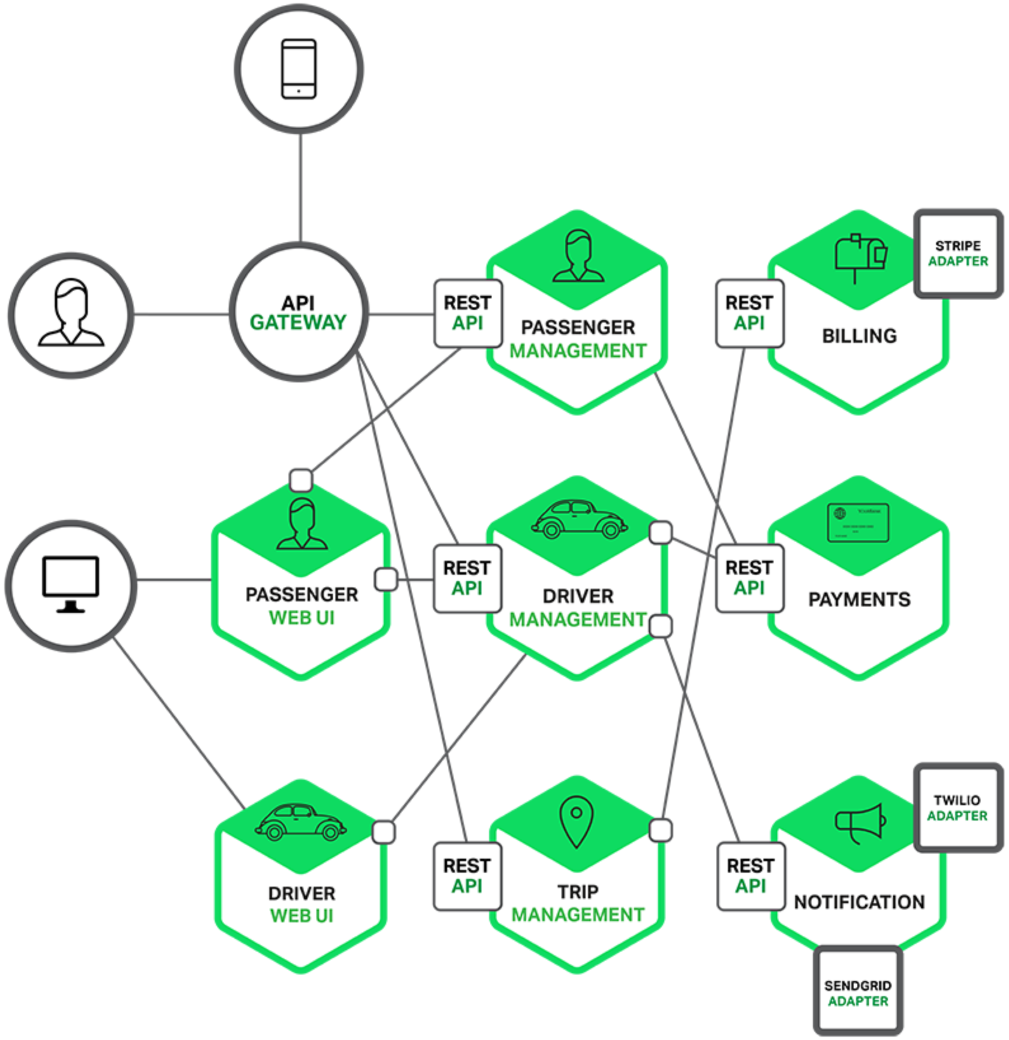
FROM Sets the Base Image for subsequent instructions.
RUN execute any commands in a new layer on top of the current image and commit the results.
CMD provide defaults for an executing container.
EXPOSE informs Docker that the container listens on the specified network ports at runtime.
ENV sets environment variable.
COPY copies new files or directories to container.
ENTRYPOINT configures a container that will run as an executable.
VOLUME creates a mount point for externally mounted volumes or other containers.
USER sets the user name for following RUN / CMD / ENTRYPOINT commands.
WORKDIR sets the working directory.
ONBUILD adds a trigger instruction when the image is used as the base for another build.
LABEL apply key/value metadata to your images, containers, or daemons.

Docker: syntactic structure

```
FROM ubuntu

RUN apt-get update && apt-get install -y postgresql-9.3 \
    postgresql-client-9.3 postgresql-contrib-9.3

USER postgres
RUN /etc/init.d/postgresql start && \
    psql --command "CREATE USER docker WITH SUPERUSER PASSWORD 'docker';" && \
    createdb -O docker docker
RUN echo "host all  all    0.0.0.0/0  md5" >> \
    /etc/postgresql/9.3/main/pg_hba.conf
RUN echo "listen_addresses='*'" >> \
    /etc/postgresql/9.3/main/postgresql.conf

EXPOSE 5432
VOLUME  ["/etc/postgresql", "/var/log/postgresql", "/var/lib/postgresql"]
CMD ["/usr/lib/postgresql/9.3/bin/postgres", "-D", \
    "/var/lib/postgresql/9.3/main", "-c", \
    "config_file=/etc/postgresql/9.3/main/postgresql.conf"]
```

Docker: microservice architecture

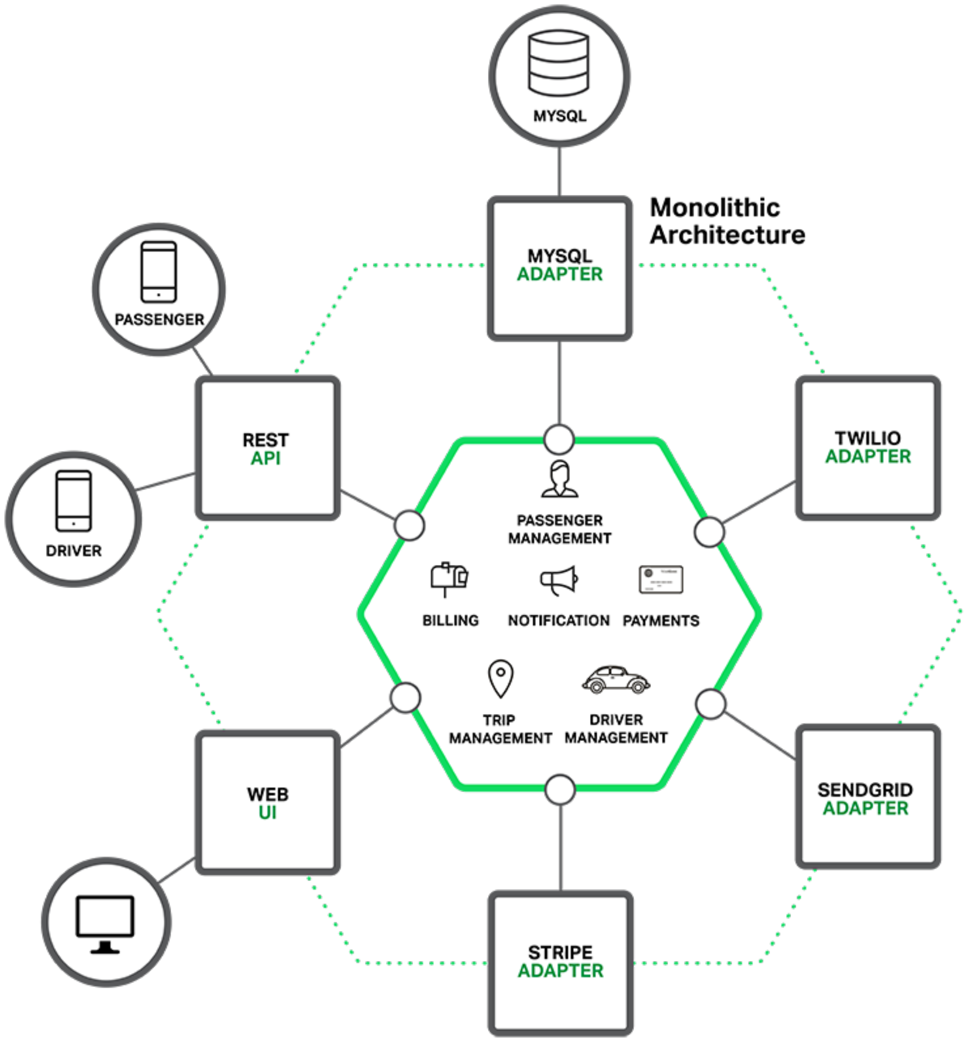
↳ Loosely coupled services with bounded contexts



nginx.com

Docker: microservice architecture

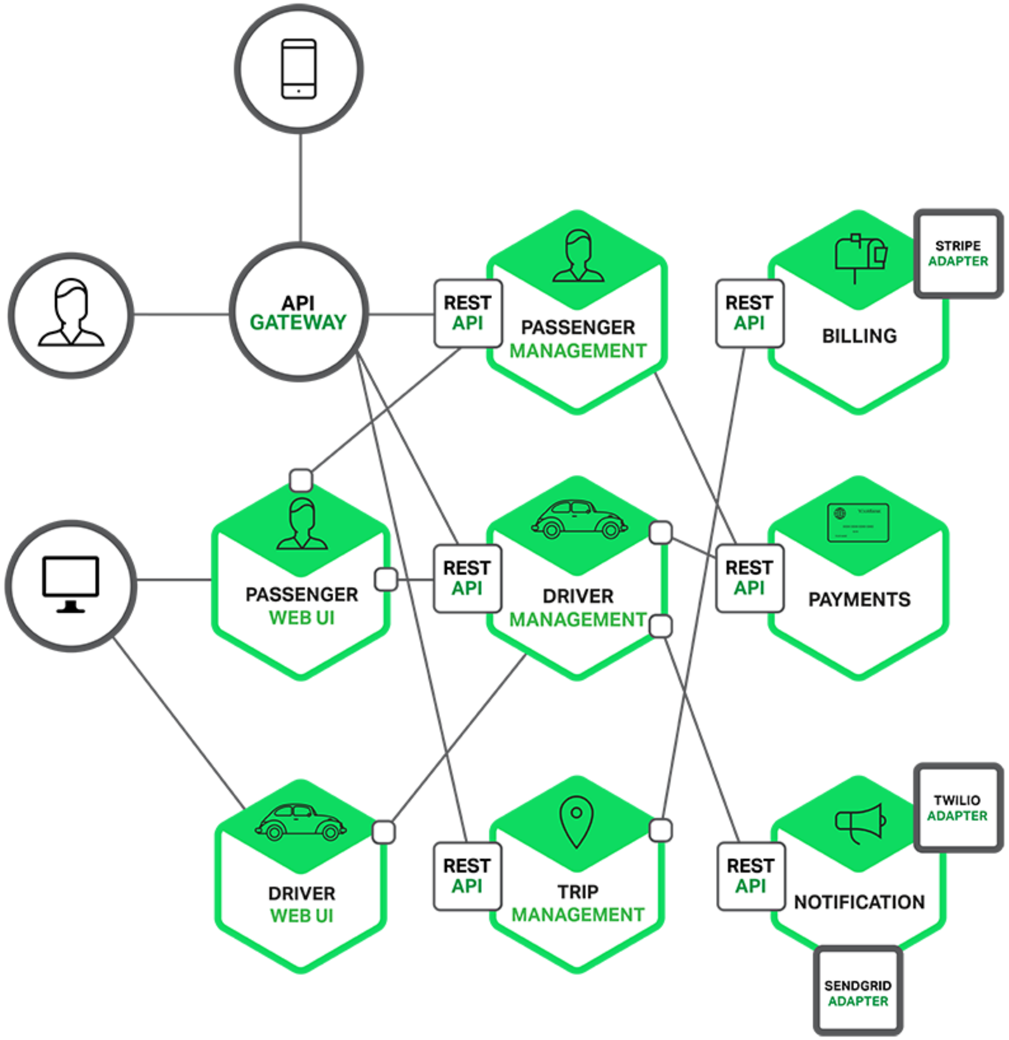
↳ Loosely coupled services with bounded contexts



nginx.com

Docker: microservice architecture

↳ Loosely coupled services with bounded contexts



nginx.com

Docker: orchestration

Connect services using rules declared in an orchestration file also in the SCM

```
version: '2'

services:
   db:
     image: mysql:5.7
     volumes:
       - "./.data/db:/var/lib/mysql"
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: wordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     links:
       - db
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_PASSWORD: wordpress
```

See 12 Factor App, rule #3

Port mapping host:container

Service connection

Docker: Ansible

```
- hosts: django
  roles:
    - django-gunicorn
- hosts: gulp
  roles:
    - gulp-static
- hosts: nginx
  roles:
    - role: j00bar.nginx-container
      ASSET_PATHS:
        - /tmp/django/static/
        - /tmp/gulp/node/dist/
```

Docker: Ansible

```
- hosts: django
  roles:
    - django-gunicorn
- hosts: gulp
  roles:
    - gulp-static
- hosts: nginx
  roles:
    - role: j00bar.ngi
      ASSET_PATHS:
        - /tmp/django/
        - /tmp/gulp/nc
```

```
django:
  image: centos:7
  environment:
    DATABASE_URL: "pgsql://{{ POSTGRES_USER }}:{{ POSTGRES_PASSWORD
}}@postgresql:5432/{{ POSTGRES_DB }}"
  expose:
    - "{{ DJANGO_PORT }}"
  working_dir: "{{ DJANGO_ROOT }}"
  links:
    - postgresql
  user: "{{ DJANGO_USER }}"
  command: ['/usr/bin/dumb-init', '{{ DJANGO_VENV }}/bin/gunicorn', '-w', '2',
'-b', '0.0.0.0:{{ DJANGO_PORT }}', 'example.wsgi:application']
  dev_overrides:
    command: ['/usr/bin/dumb-init', '{{ DJANGO_VENV }}/bin/python',
'manage.py', 'runserver', '0.0.0.0:{{ DJANGO_PORT }}']
    volumes:
      - "$PWD:{{ DJANGO_ROOT }}"
  options:
    kube:
      runAsUser: 1000
```

Docker: Ansible

```
- hosts: django
  roles:
    - django-gunicorn
- hosts: gulp
  roles:
    - gulp-static
- hosts: nginx
  roles:
    - role: j00bar.ngi
      ASSET_PATHS:
        - /tmp/django/
        - /tmp/gulp/nc
```

```
gulp:
  image: centos:7
  user: {{ NODE_USER }}
  command: /bin/false
  dev_overrides:
    working_dir: "{{ NODE_HOME }}"
    command: ['/usr/bin/dumb-init', '{{ NODE_ROOT }}/node_modules/.bin/gulp']
    ports:
      - "80:{{ GULP_DEV_PORT }}"
    volumes:
      - "$PWD:{{ NODE_HOME }}"
    links:
      - django
  options:
    kube:
      state: absent
```

Docker: Ansible

```
- hosts: django
  roles:
    - django-gunicorn
- hosts: gulp
  roles:
    - gulp-static
- hosts: nginx
  roles:
    - role: j00bar.ngi
      ASSET_PATHS:
        - /tmp/django/
        - /tmp/gulp/nc
```

```
nginx:
  image: centos:7
  ports:
    - "80:{{ DJANGO_PORT }}"
  user: 'nginx'
  links:
    - django
  command: ['/usr/bin/dumb-init', 'nginx', '-c', '/etc/nginx/nginx.conf']
  dev_overrides:
    ports: []
    command: '/bin/false'
  options:
    kube:
      runAsUser: 997
```

Jenkins 2 Pipeline can use Docker agents

Agents in pipeline-model-definition can be hosts or Docker containers

```
pipeline {
  agent docker:'node:6.3'
  stages {
    stage('build') {
      steps {
        sh 'npm --version'
        sh 'npm install'
        sh 'npm test'
      }
    }
  }
}
```

Coming soon in version 0.6: from [JENKINS-39216] a new parameter for `agent` to auto-.build a Dockerfile and run the build in a container based on that image

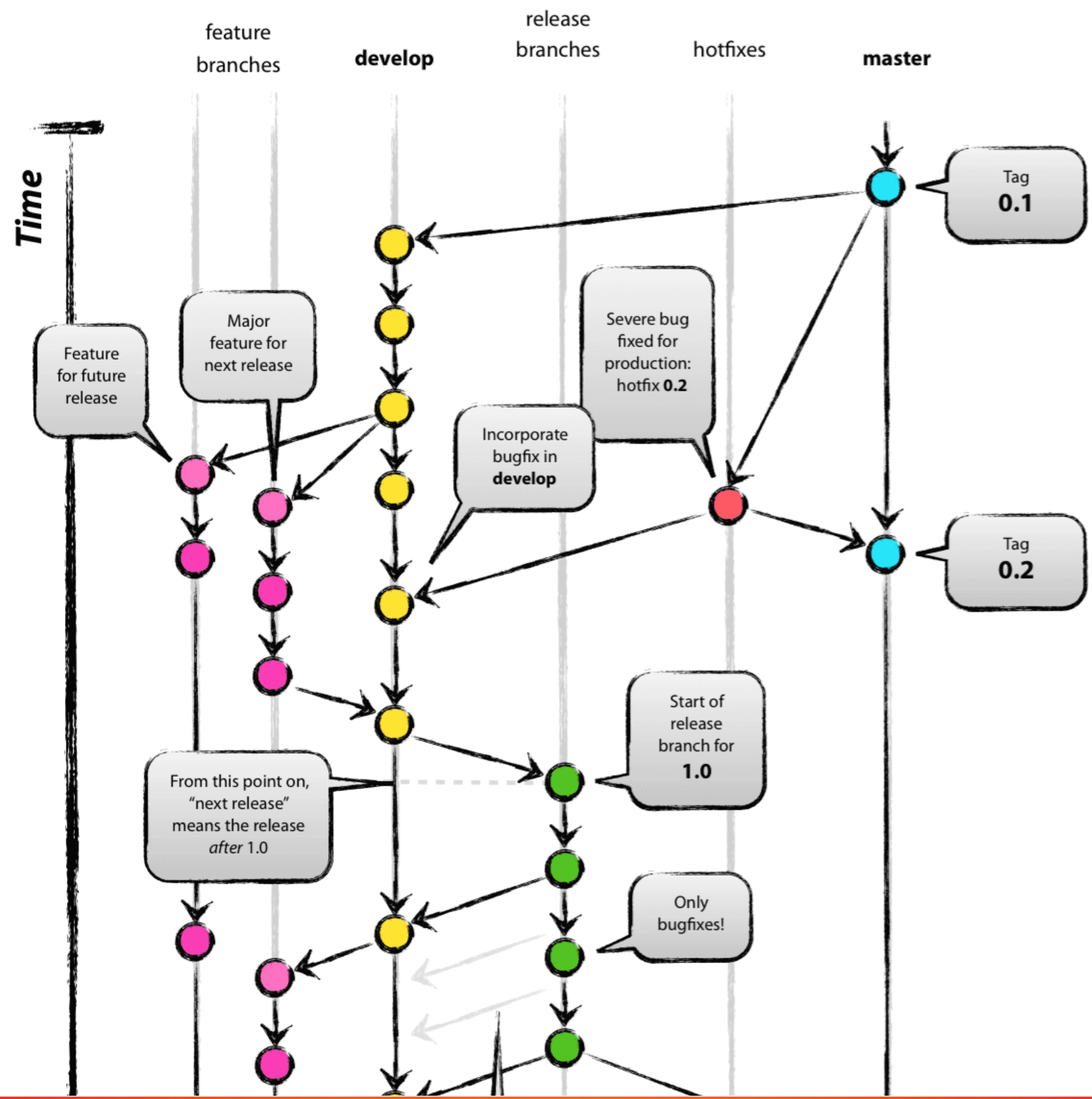
Take home message

Infrastructure as code: record it in the SCM

- Software component
  - - Source code and dependencies (`Makefile`/`CMakeLists.txt`/`package.json`/…)
  - - Instructions to build, test and deploy (`Jenkinsfile`)
- Container
  - - Instructions to configure environment for each service (`Dockerfile`)
  - - Instructions to link services together (`docker-compose.yml`/`answers.conf`/…)

Use the same SCM workflows that we use for software

- Branches (master, hotfix, development, feature)
- Code reviews, issues, and collaboration tools →→→ traceability
- Versioning, logs
- **No dark matter**


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!