Software Engineering Conference Russia

October 2017, St. Petersburg

SECR

Как казаки код двигали

Владимир Трубников

Principal SW Engineer

DELL EMC

О проекте

- Backend(s)
- Middleware
- Library
- Frontend
- NodeJS part of frontend
- Additional modules (mailing service, integration with other applications, etc)

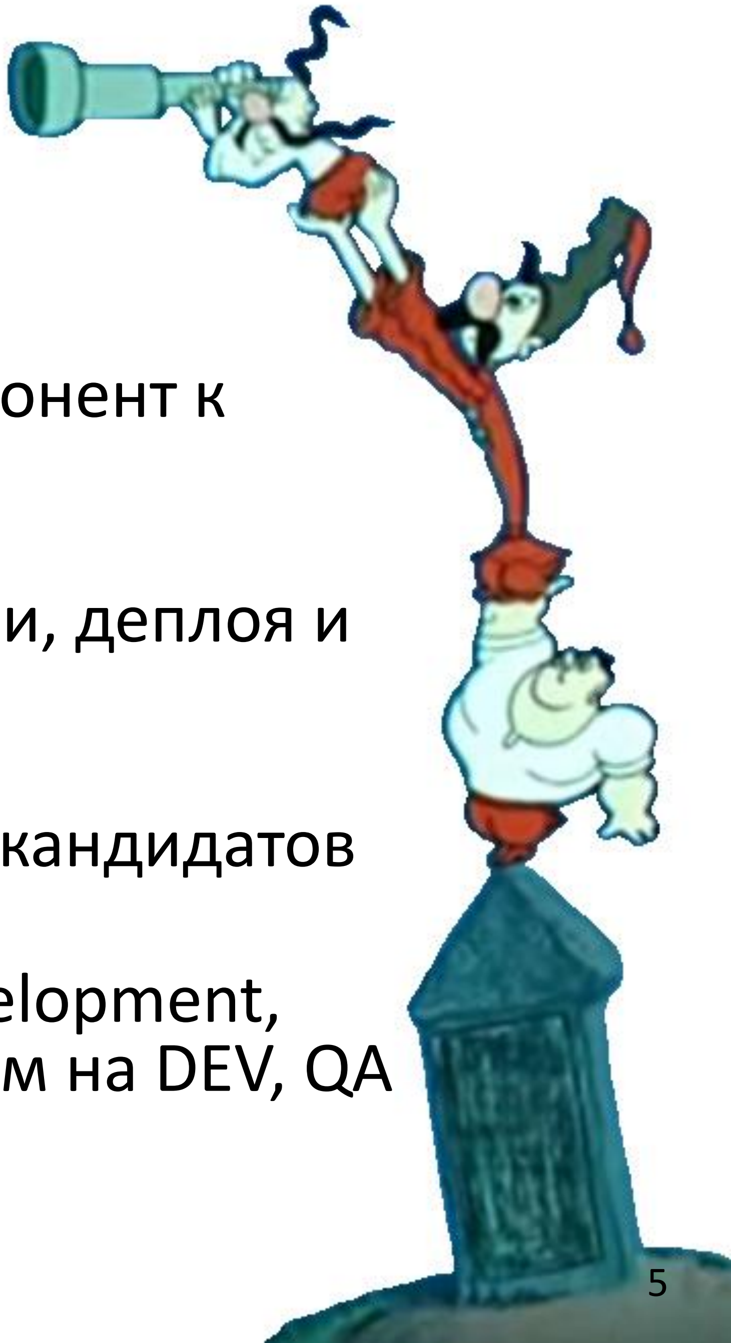
Начало пути

- Manual (local) builds (No Jenkins)
- No artifactory
- No tests
- Deploy to PROD and fixing issues on PROD directly during weekend

Проблемы

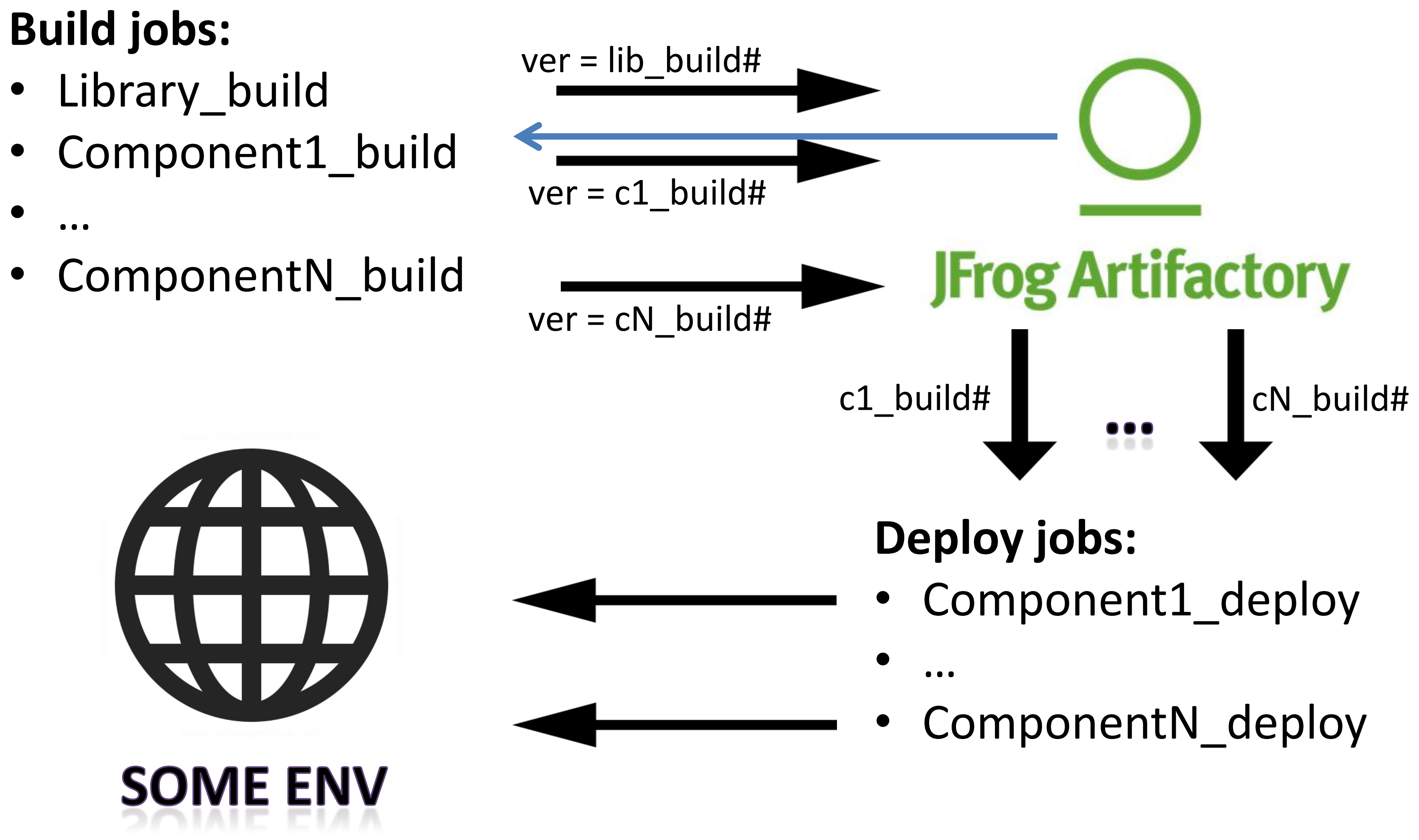
- Деплой несогласованных версий
- Разные версии внутренних библиотек
- Built ≠ Tested ≠ Deployed
- Нерелевантные configs
- Deployed version ≠ deployed version (из-за частых апдейтов и hot-фиксов)

Цель



- Сквозная нумерация версий
- Четкое соответствие версий компонент к общей версии
- Автоматическая процедура сборки, деплоя и тестирования в 1 клик
- Отдельное хранилище для релиз кандидатов
- Соответствие между кодом в development, integration, master бранчах с кодом на DEV, QA и PROD

Подготовительный этап. Build&Deploy.



Build jobs:

- Library_build
- Component1_build
- …
- ComponentN_build

Deploy jobs:

- Component1_deploy
- …
- ComponentN_deploy

Подготовительный этап. Tests.

Test jobs:

- TestSuite1_job
- …
- TestSuiteM_job

ver = ts1_build#

ver = tsM_build#

JFrog Artifactory

QATool
Tool for storing, visualization and analysis of tests results

Настройка процесса

Шаг 1:
git branches:
- feature/fix branches
- development = DEV
- integration = QA
- master = PROD

Шаг 2:
переменные Jenkins:
- major_version
- minor_version
- patch_version

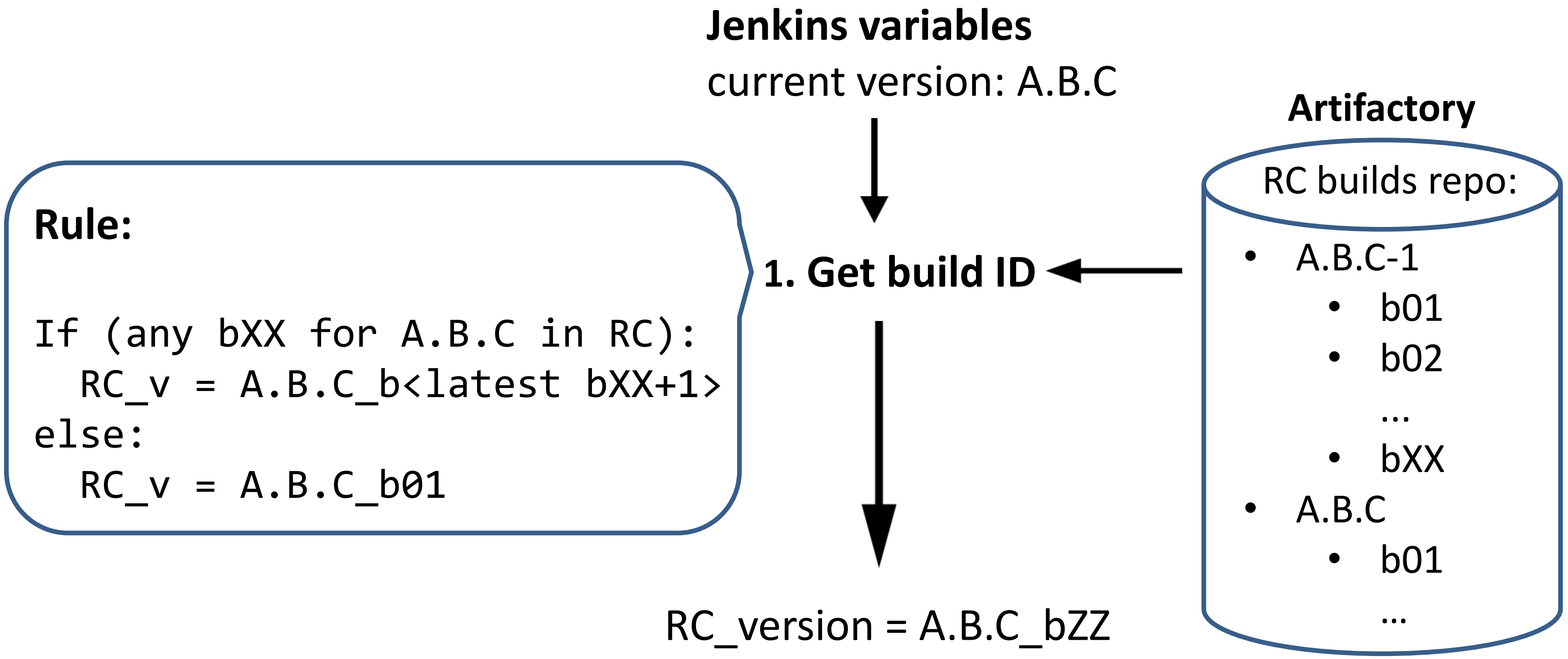
Шаг 3:
Отдельная репозитория в artifactory для хранения релиз кандидатов

Краткое описание шагов.

Используя Jenkins Pipeline job with Groovy:

1. Get build ID (имея major, minor & patch версии вычисляем номер следующего RC)
2. Build all
3. Deploy all
4. Test all
5. Copy RC to artifactory
6. Merge to next branch

Шаг 1: Вычисление версии.



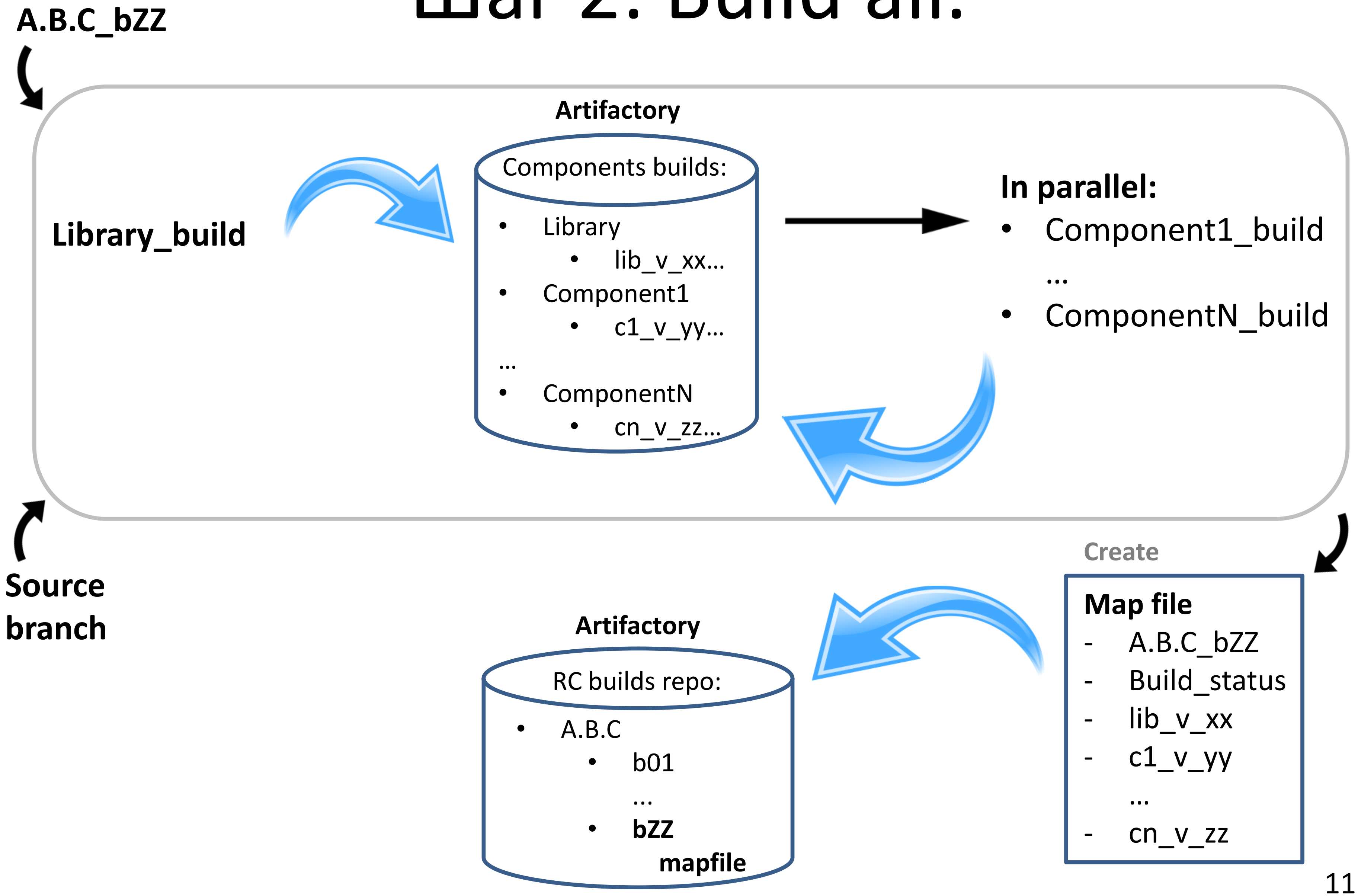
Rule:

```
If (any bXX for A.B.C in RC):
  RC_v = A.B.C_b<latest bXX+1>
else:
  RC_v = A.B.C_b01
```

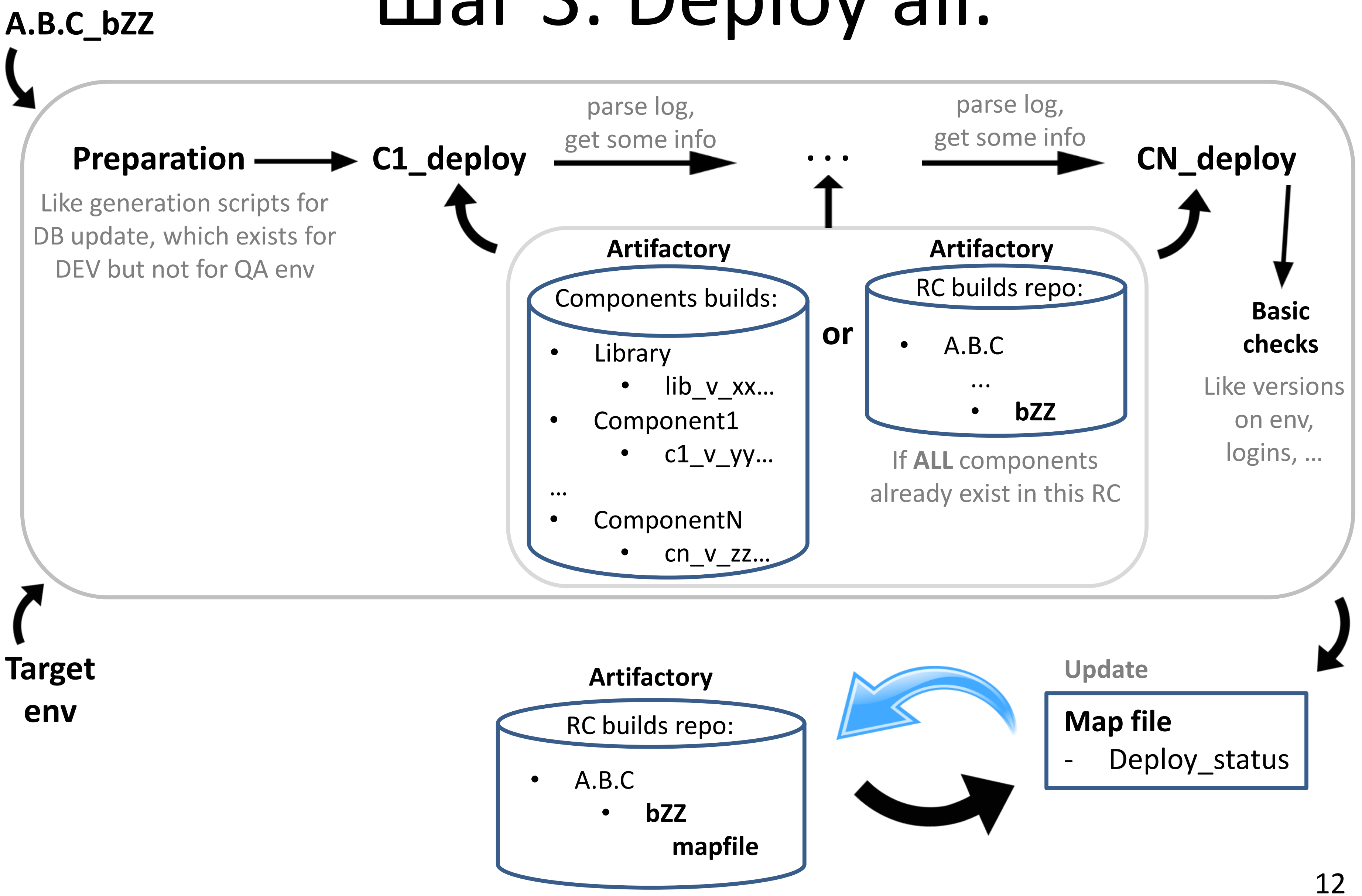
In Groovy via cURL:

```
def response = sh(
    script: "curl -s -S -k -u user:key -X
        GET repo_host/api/storage/repo_name/A.B.C?list&deep=0&listFolders=1",
    returnStdout: true).trim()
```

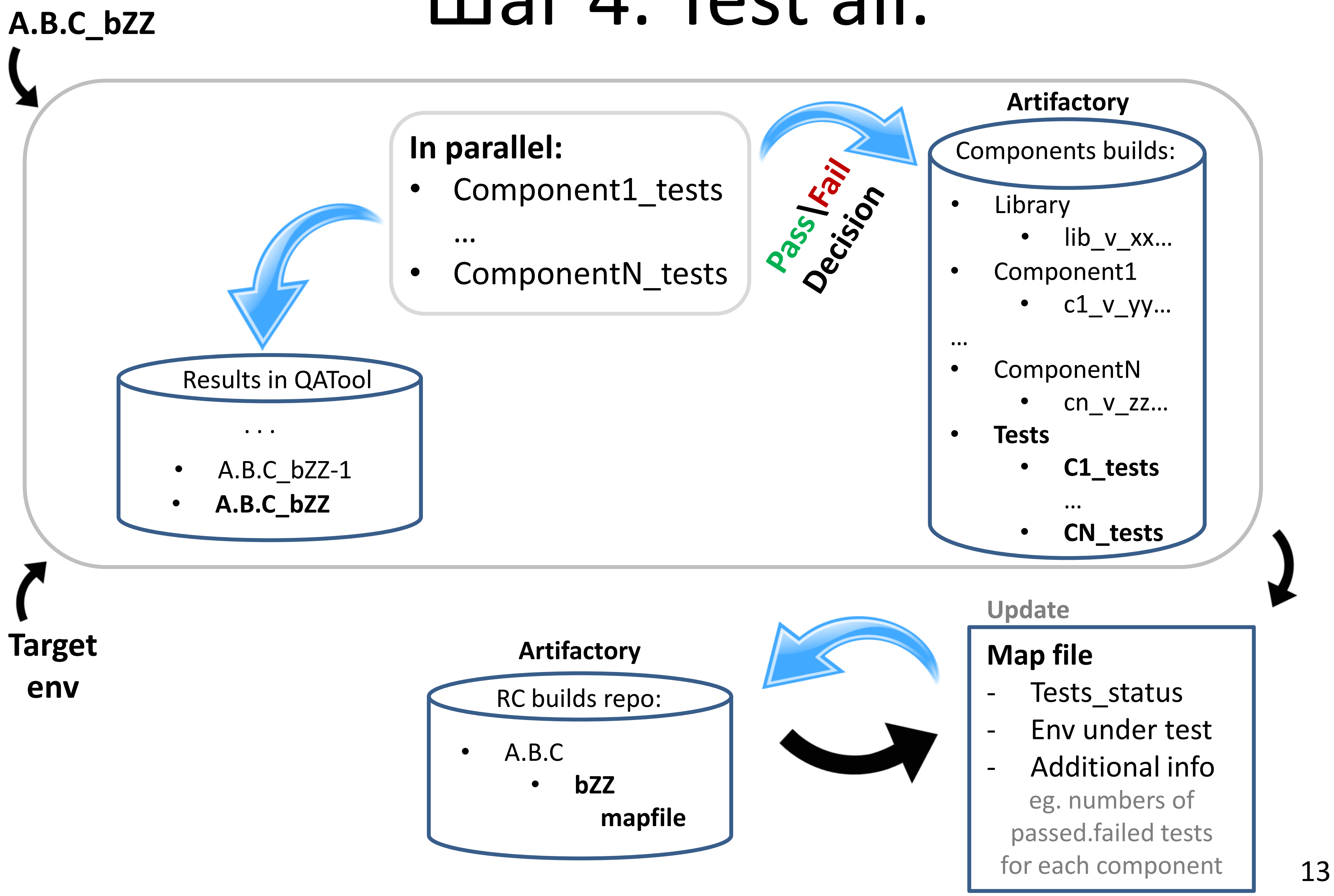
Шаг 2: Build all.



Шаг 3. Deploy all.



Шar 4. Test all.



In parallel:
- Component1_tests
…
- ComponentN_tests

Results in QATool

. . .
- A.B.C_bZZ-1
- **A.B.C_bZZ**

Artifactory

Components builds:
- Library
  - lib_v_xx…
- Component1
  - c1_v_yy…

…
- ComponentN
  - cn_v_zz…
- **Tests**
  - **C1_tests**

    …
  - **CN_tests**

Artifactory

RC builds repo:
- A.B.C
  - **bZZ**

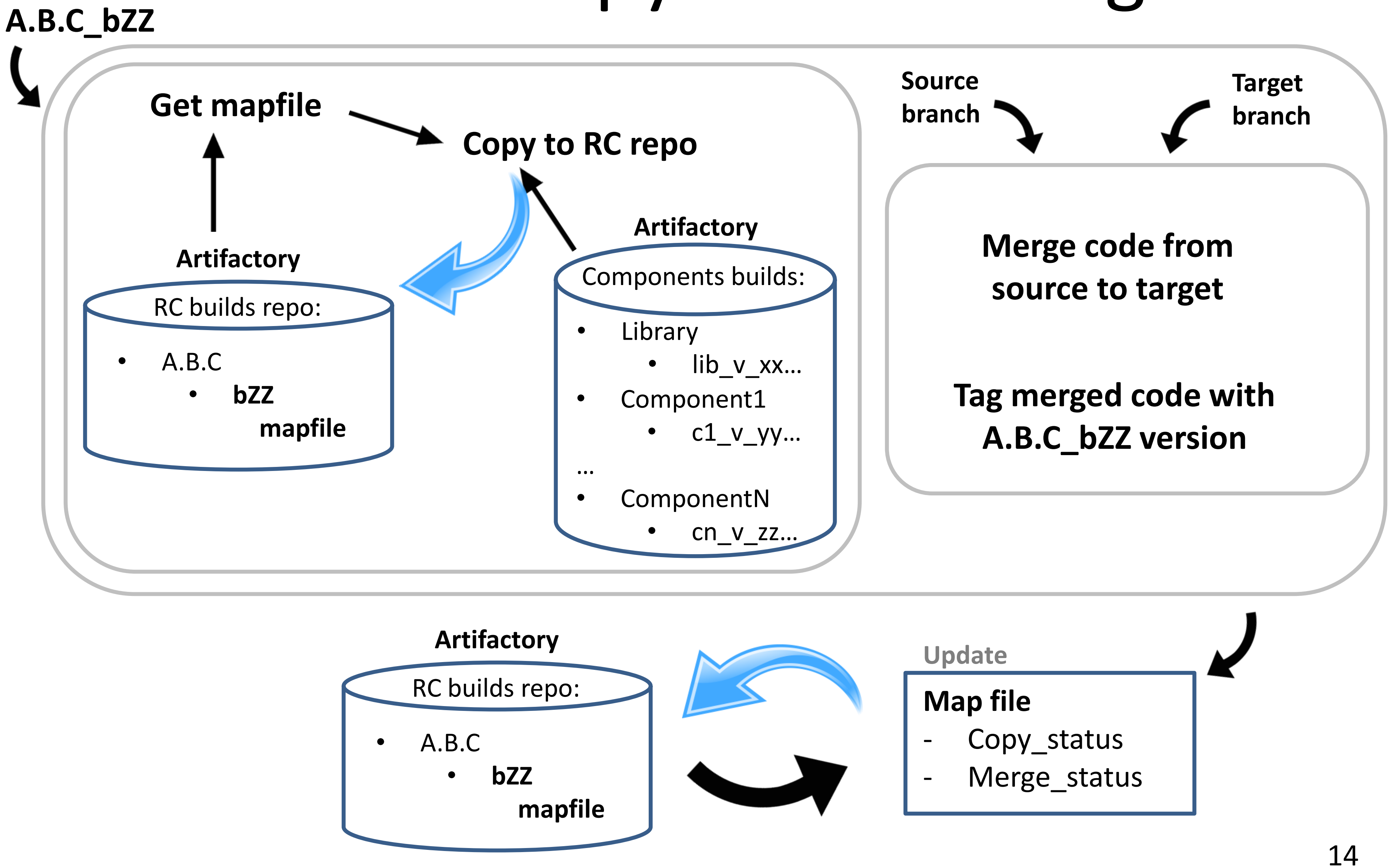
    mapfile

Update

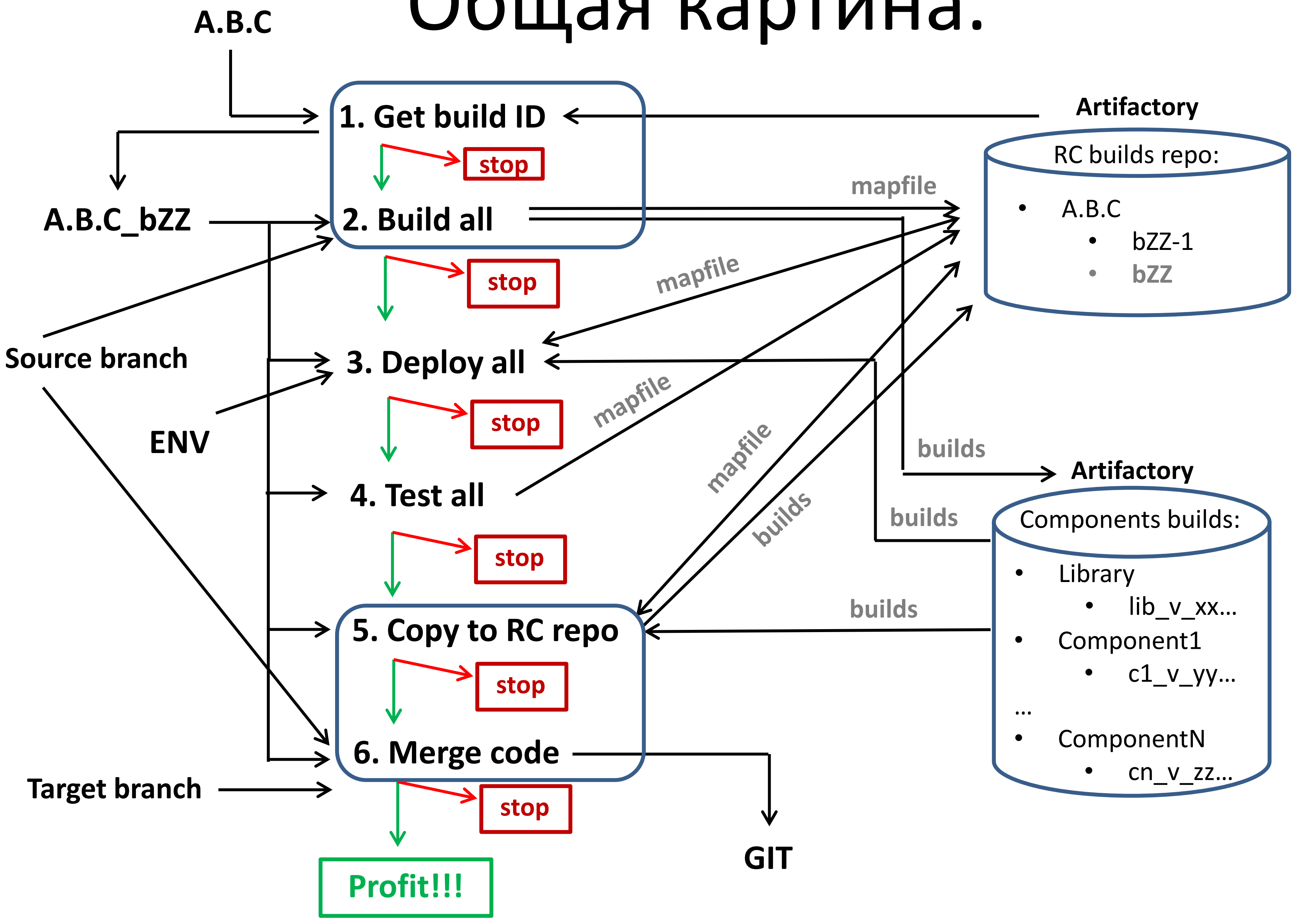
Map file
- Tests_status
- Env under test
- Additional info

eg. numbers of passed.failed tests for each component

Шаг 5-6. Copy RC and merge.



Общая картина.



Итоги 1

Стало так

Было как-то так…

1. Build version

1.1. Calculate version

.2. Build Library

..3. Build Components

- **Build C1**
- **…**
- **Build CN**

1. Deploy Components

- **Deploy C1**
- **…**
- **Deploy CN**

3. Test all

3.1. Execute tests

- **Test suite 1**
- **…**
- **Test suite M**

4. Copy & merge

4.1. Copy to RC artifactory
4.2. Merge branches

Построение RC в 1 клик

- ~ 1000 строк кода groovy
- > 20 jobs
- авто-версионирование
- RC по расписанию

Итоги 1 (примечание)

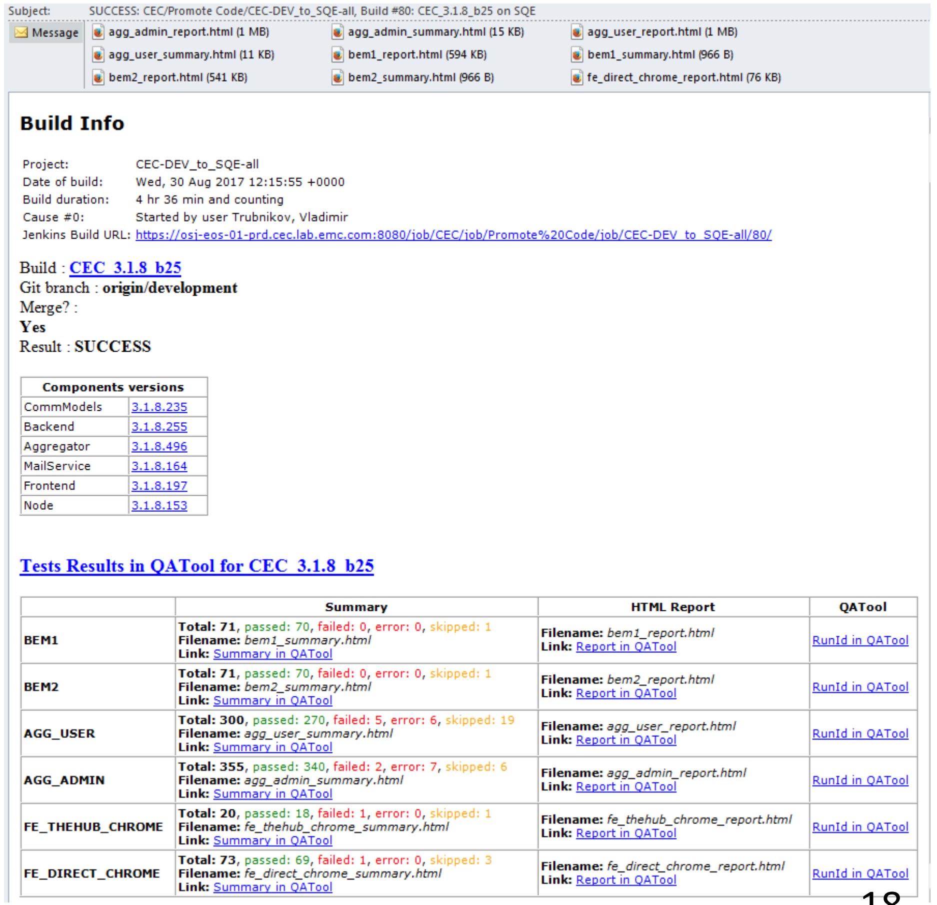
Sandbox:

- Разрешены только базовые команды
- Возможность добавить только при получении not approved exception
- Возможность добавления комманд только по одной
- Скрытие вызовов в try/catch

Итоги 2

- Groovy + Jenkins = NICE!
  - Не забываем про правила (и их backups!)
  - Скрипты в git (не в Jenkins)
  - Шарим инфу через artifactory
  - 1 версия для всех
- Легко расширяемо

- Красивые письма



Subject: SUCCESS: CEC/Promote Code/CEC-DEV_to_SQE-all, Build #80: CEC_3.1.8_b25 on SQE

Message | agg_admin_report.html (1 MB) | agg_admin_summary.html (15 KB) | agg_user_report.html (1 MB) | agg_user_summary.html (11 KB) | bem1_report.html (594 KB) | bem1_summary.html (966 B) | bem2_report.html (541 KB) | bem2_summary.html (966 B) | fe_direct_chrome_report.html (76 KB)

Build Info

Project:	CEC-DEV_to_SQE-all
Date of build:	Wed, 30 Aug 2017 12:15:55 +0000
Build duration:	4 hr 36 min and counting
Cause #0:	Started by user Trubnikov, Vladimir
Jenkins Build URL:	https://osj-eos-01-prd.cec.lab.emc.com:8080/job/CEC/job/Promote%20Code/job/CEC-DEV_to_SQE-all/80/

Build : CEC_3.1.8_b25
Git branch : **origin/development**
Merge? :
Yes
Result : **SUCCESS**

Components versions	
CommModels	3.1.8.235
Backend	3.1.8.255
Aggregator	3.1.8.496
MailService	3.1.8.164
Frontend	3.1.8.197
Node	3.1.8.153

Tests Results in QATool for CEC_3.1.8_b25

	Summary	HTML Report	QATool
BEM1	**Total: 71**, passed: 70, failed: 0, error: 0, skipped: 1 **Filename:** *bem1_summary.html* **Link:** Summary in QATool	**Filename:** *bem1_report.html* **Link:** Report in QATool	RunId in QATool
BEM2	**Total: 71**, passed: 70, failed: 0, error: 0, skipped: 1 **Filename:** *bem2_summary.html* **Link:** Summary in QATool	**Filename:** *bem2_report.html* **Link:** Report in QATool	RunId in QATool
AGG_USER	**Total: 300**, passed: 270, failed: 5, error: 6, skipped: 19 **Filename:** *agg_user_summary.html* **Link:** Summary in QATool	**Filename:** *agg_user_report.html* **Link:** Report in QATool	RunId in QATool
AGG_ADMIN	**Total: 355**, passed: 340, failed: 2, error: 7, skipped: 6 **Filename:** *agg_admin_summary.html* **Link:** Summary in QATool	**Filename:** *agg_admin_report.html* **Link:** Report in QATool	RunId in QATool
FE_THEHUB_CHROME	**Total: 20**, passed: 18, failed: 1, error: 0, skipped: 1 **Filename:** *fe_thehub_chrome_summary.html* **Link:** Summary in QATool	**Filename:** *fe_thehub_chrome_report.html* **Link:** Report in QATool	RunId in QATool
FE_DIRECT_CHROME	**Total: 73**, passed: 69, failed: 1, error: 0, skipped: 3 **Filename:** *fe_direct_chrome_summary.html* **Link:** Summary in QATool	**Filename:** *fe_direct_chrome_report.html* **Link:** Report in QATool	RunId in QATool

Q&A?

Подвал

схемы
Groovy скриптов

Шаг 2. Implementation

```
try {
   stage("Build Library") {
      Lib_build = build job: ‘Library_build', parameters: […]
      Lib_version = CM_build.getNumber()
   }
   …
   parallel_builds = [:]
   parallel_builds[“C1"] = {  stage("Build C1") {
         C1_build = build job: ‘Component1_build', parameters: [Lib_version, …]
         C1_version = BEM_build.getNumber()
   }}
   …
   parallel parallel_builds
} finally { stage("Create map file") {
      properties += "BUILD_version=A.B.C_bZZ“
      properties += “C1_v=${Lib_version}“
      …
      properties += "BUILD_STATUS=${currentBuild.result}"

      writeFile file: "mapfile", text: "${properties}"

      sh "curl -s -S -u user:key -X PUT repo_host/A.B.C/bZZ/mapfile -T mapfile"
}}
```

Шаг 3. Implementation

```
try {
   stage(“Preparation") {…}

   stage(“Select repo with binaries") {
      res = sh ( script: "curl GET mapfile", returnStdout: true).trim()  }

   stage("Deploy C1") {
      deploy_C1 = build job: ‘deploy_C1', parameters: [C1_v_yy, env]
      log = getItemByFullName("deploy_C1").getBuildByNumber(deploy_C1.getNumber()).logFile
      assert !log.contains("DB_ERROR"), "Error on upgrading C1 DB"  … }

   stage(“Basic checks (eg. verify login to s/w)") {
      body = """{"credentials": {"username": “usr", "password": “pass"}}"""
      test = httpRequest httpMode: 'POST', url: 'http://endpoint',requestBody: body
      assert test.status == 200, "Didn't logged in successfully“ …  }

} finally {
      res = sh ( script: "curl GET … mapfile", returnStdout: true).trim()
      mapfile_content = readProperties text: "${res}“

      mapfile_content += "DEPLOY_STATUS=${currentBuild.result}"
      writeFile file: "mapfile", text: "${mapfile_content}“
      sh "curl PUT repo/A.B.C/bZZ/mapfile -T mapfile"
}
```

Шаг 4. Implementation

```
try {
   all_in_parallel = [:]
   all_in_parallel[“C1_Tests"] = { stage("Test C1") {
   test_C1 = build job: Component1_tests, parameters: [RC_ver, env, …]
   sh "wget  results.tar -d --header='X-JFrog-Art-Api: key' path_to_results_in_artifactory "
   sh 'tar -xvf results.tar '

   step([$class    : 'XUnitBuilder', testTimeMargin: '3000', thresholdMode: 2,
       thresholds: [
                        [$class: 'FailedThreshold', failureThreshold: '10', unstableThreshold: ''],
                        [$class: 'SkippedThreshold', failureThreshold: '', unstableThreshold: '']],
       tools: [$class: 'JUnitType', pattern: 'results/**/*.xml', skipNoTestFiles: true]
    ]
    …
    parallel all_in_parallel

} finally {
      res = sh ( script: "curl GET … mapfile", returnStdout: true).trim()
      mapfile_content = readProperties text: "${res}“

      mapfile_content += “TESTS_STATUS=${currentBuild.result}“
      …
      writeFile file: "mapfile", text: "${mapfile_content}“
      sh "curl PUT repo/A.B.C/bZZ/mapfile -T mapfile"
}
```

Шаг 5-6. Implementation

```
try {
  stage("Generate paths for copying") {
     res = sh ( script: "curl GET repo/A.B.C/bZZ/mapfile ", returnStdout: true).trim()
     <create  source_paths_list>
     <create  target_paths_list>     }
  stage("Copy artifacts to int repo") {
     for (comp : source_paths) {
        response = sh ( script: "curl POST ${comp.value}?to=${getItemByKey(target_paths, comp.key).value}",
                               returnStdout: true).trim()
        assert response ==“OK”, "Got issue on copying artifacts.“   }
  stage("Merge branches") {
      withCredentials( [credentialsId: '8fbcf128-5630-4bb8-8f86-d060d1d0d31e, …] )  {
          for (def git_repo : git_repos) {
              sh "git clone https://user:pass@$git_repo"
              dir(git_repo) {
                  sh "git checkout ${source_branch}"
                  sh "git checkout ${target_branch}"
                  sh "git merge ${s_branch_name}"
                  sh "git push origin ${t_branch_name}"

                  sh "git tag ${build_version}"
                  sh "git push origin ${build_version}"   }}}}
} finally {
      // update mapfile
}
```