Лекция 12: Непрерывное развёртывание

Юрий Литвинов
y.litvinov@spbu.ru

19.05.2026

Выпуск новых версий

- Как быстро изменения в одну строчку кода попадают к пользователям?
- Сколько для этого требуется усилий?
- Насколько это процесс повторяем?
- Сколько человек вовлечено в этот процесс?

Антипаттерн управления релизами 1:

Ручное развёртывание

- Необходимость подробного документирования процесса
  - Или эксперта
- Повышение стоимости и времени развёртывания
  - Дополнительная нагрузка на специалистов
- Непредсказуемость процесса
  - Отличия в конфигурации приложений или окружений
- Отсутствие гарантий надёжности и повторяемости

Антипаттерн управления релизами 2:

Развёртывание только после завершения разработки

- Все ошибки проявляются впервые
  - Включая неверные предположения о системе и её окружении
- Специальная команда из разных специалистов
  - Между которыми нужно наладить взаимодействие
  - “Оторванность” разработчиков от реальной жизни
- Попытки срочных “горячих изменений”

Антипаттерн управления релизами 3:

Ручное управление окружением в продакшене

- Ручная подготовка окружения
  - Документация, эксперты, ...
- Отсутствие повторяемости процесса размещения
  - Сложно/невозможно откатиться к предыдущей (стабильной) версии
- Различия в окружениях разработки и продакшена
  - Фиксы багов
  - Изменения настроек БД, серверов и т.п.
  - Переменные окружения
  - ...
- Никто толком не знает, что сейчас в продакшене

Решение: частые автоматизированные релизы

- Быстрая обратная связь
  - Обратная связь от каждого изменения
  - Получается как можно быстрее
  - Оперативное реагирование
- Повышение гибкости процесса развёртывания
- Снижение количества ошибок
- Снижение уровня стресса
- Расширение возможностей команды
- Снижение порога вхождения в проект

Принципы непрерывного развёртывания

- Повторяемый, надёжный процесс выпуска версий
- Максимальная автоматизация
- Максимальное версионирование всего
- Если где-то есть проблема, делаем это как можно раньше и чаще
- “Сделано” значит “попало в релиз”
- Выпуск версий — общая ответственность
- Постоянное улучшение

Конфигурационное управление

- Использование систем контроля версий
- Управление зависимостями
  - Сторонние библиотеки и внутренние зависимости
- Управление конфигурациями ПО
  - Конфигурации и гибкость
  - Типы конфигураций
- Управление окружением ПО
  - Автоматизация создания и настройки

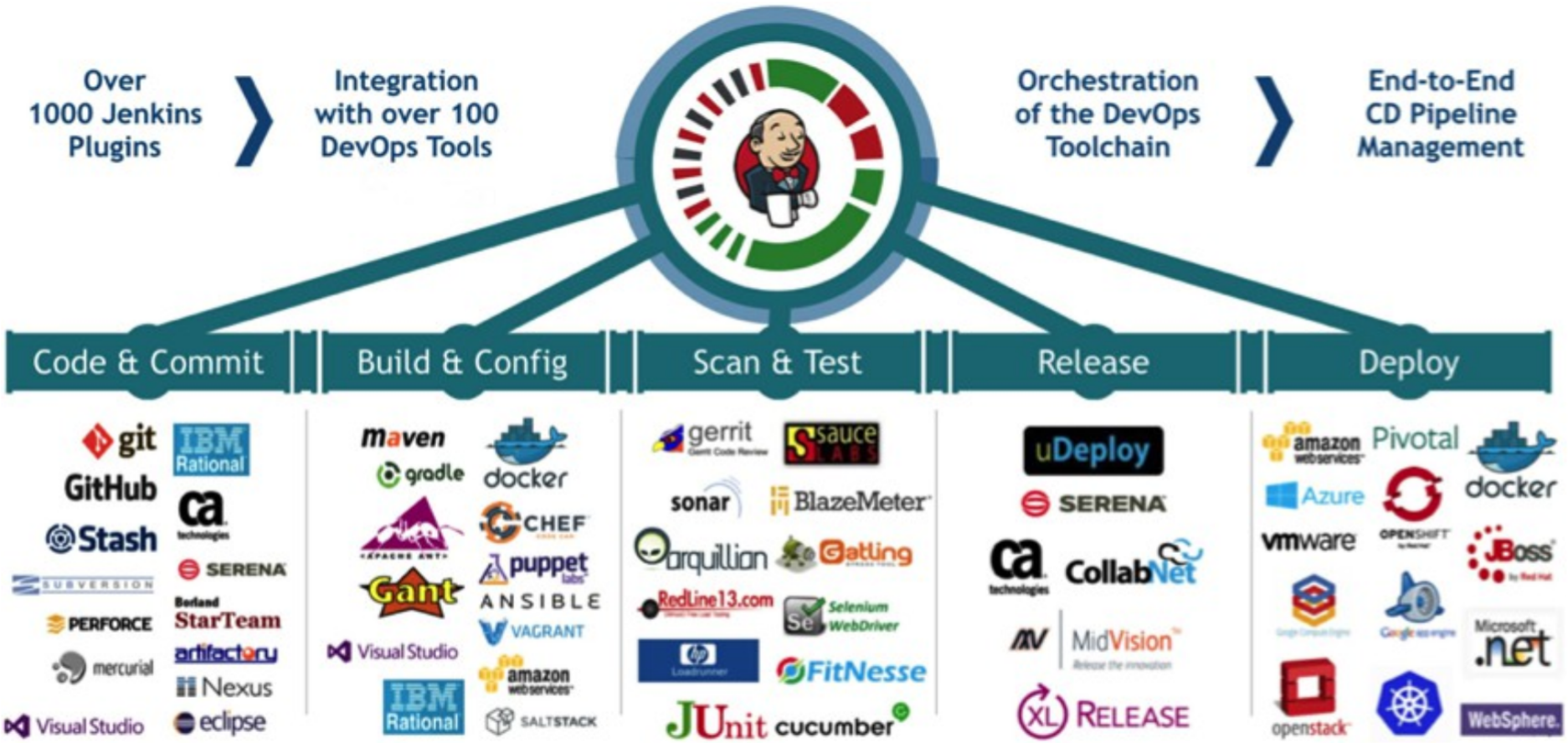
Continuous Integration

- Предусловия
  - Система контроля версий
  - Автоматизированный процесс сборки
  - Соглашение внутри команды
- Необходимые практики
  - Регулярные коммиты (и мерджи)
  - Вменяемая система автотестов
  - Вменяемый по времени процесс сборки и тестирования

TeamCity


IntegrationBuild (FreeBSD)
Pending (3)
1 queued
Run ...
#8059 Updating sources
No artifacts
Changes (7)
2h:34m left
Stop
#8058 Tests failed: 2 (1 new), passed: 13658, ignored: 133, muted: 5
No artifacts
Dmitry Neverov (1)
2 minutes ago (2h:32m)
IntegrationBuild (HSQLDB Incremental)
Pending (4)
1 queued
Run ...
#11787 Tests passed: 2232, ignored: 12, muted: 2; Running '[TeamCity] Server tests'
No artifacts
Evgeniy Koshkin (1)
1h:12m left
Stop
#11788 Success
No artifacts
Dmitry Neverov (1)
14 minutes ago (14m:02s)
IntegrationBuild (Linux)
Pending (2)
Run ...
#7928 Tests failed: 2 (1 new), passed: 1897, ignored: 31; Running '[TeamCity] Server tests'
No artifacts
Changes (3)
1h:58m left
Stop
#7927 Tests failed: 1 (1 new), passed: 6955, ignored: 55, muted: 3; Running '[TeamCity] Server tests'
No artifacts
Evgeniy Koshkin (1)
1h:19m left
Stop
#7926 Tests passed: 12149, ignored: 75, muted: 4; Running '[TeamCity] Integration tests'
No artifacts
Changes (4)
40m:36s left
Stop
#7925 Tests failed: 1 (1 new), passed: 12303, ignored: 75, muted: 3; Running '[TeamCity] Integration tests...
No artifacts
Changes (2)
overtime: 9m:15s
Stop
#7924 Tests failed: 2 (1 new), passed: 13705, ignored: 91, muted: 4
No artifacts
Changes (3)
10 hours ago (2h:02m)
IntegrationBuild (MacOS)
Pending (3)
1 queued
Run ...
#6585 Tests failed: 1, passed: 2736, ignored: 39, muted: 3; Running '[TeamCity] Server tests'
No artifacts
Changes (3)
1h:15m left
Stop
#6584 Tests failed: 1, passed: 11901, ignored: 69, muted: 3; Running '[TeamCity] Integration tests'
No artifacts
Evgeniy Koshkin (1)
24m:50s left
Stop
#6583 Tests failed: 3 (3 new), passed: 13705, ignored: 91, muted: 3
No artifacts
Changes (5)
16 minutes ago (1h:31m)
IntegrationBuild (MS SQL)
Pending (4)
1 queued
Run ...
#6579 Tests passed: 1865, ignored: 13; Running '[TeamCity] Server tests'
No artifacts
Changes (2)
5h:58m left
Stop
#6578 Tests passed: 1280, ignored: 11; Running '[TeamCity] DB tests'
No artifacts
Changes (4)
3h:43m left
Stop

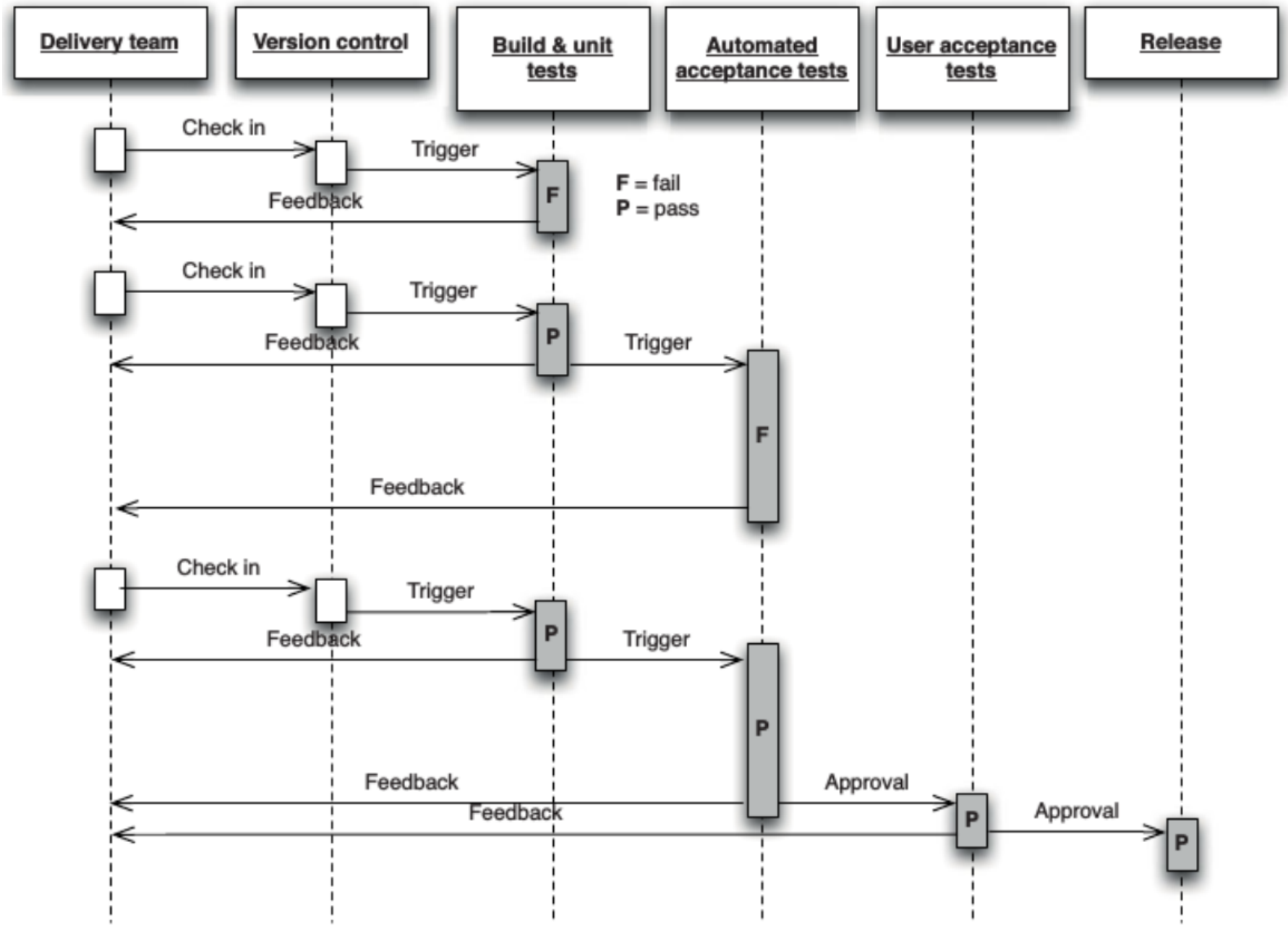
Jenkins


Over 1000 Jenkins Plugins
Integration with over 100 DevOps Tools
Orchestration of the DevOps Toolchain
End-to-End CD Pipeline Management
Code & Commit
Build & Config
Scan & Test
Release
Deploy
git
GitHub
Stash
SUBVERSION
PERFORCE
mercurial
Visual Studio
IBM Rational
ca technologies
SERENA
Borland StarTeam
artifactory
Nexus
eclipse
maven
gradle
Gant
Visual Studio
IBM Rational
docker
CHEF
puppet labs
ANSIBLE
VAGRANT
amazon web services
SALTSTACK
gerrit
Sauce Labs
sonar
BlazeMeter
Jarquillian
Gatling
RedLine13.com
Selenium WebDriver
Loadrunner
FitNesse
JUnit
cucumber
uDeploy
SERENA
ca technologies
CollabNet
MidVision
XL RELEASE
amazon web services
Pivotal
Azure
docker
vmware
OPENSHIFT
JBoss
Microsoft .net
openstack
WebSphere

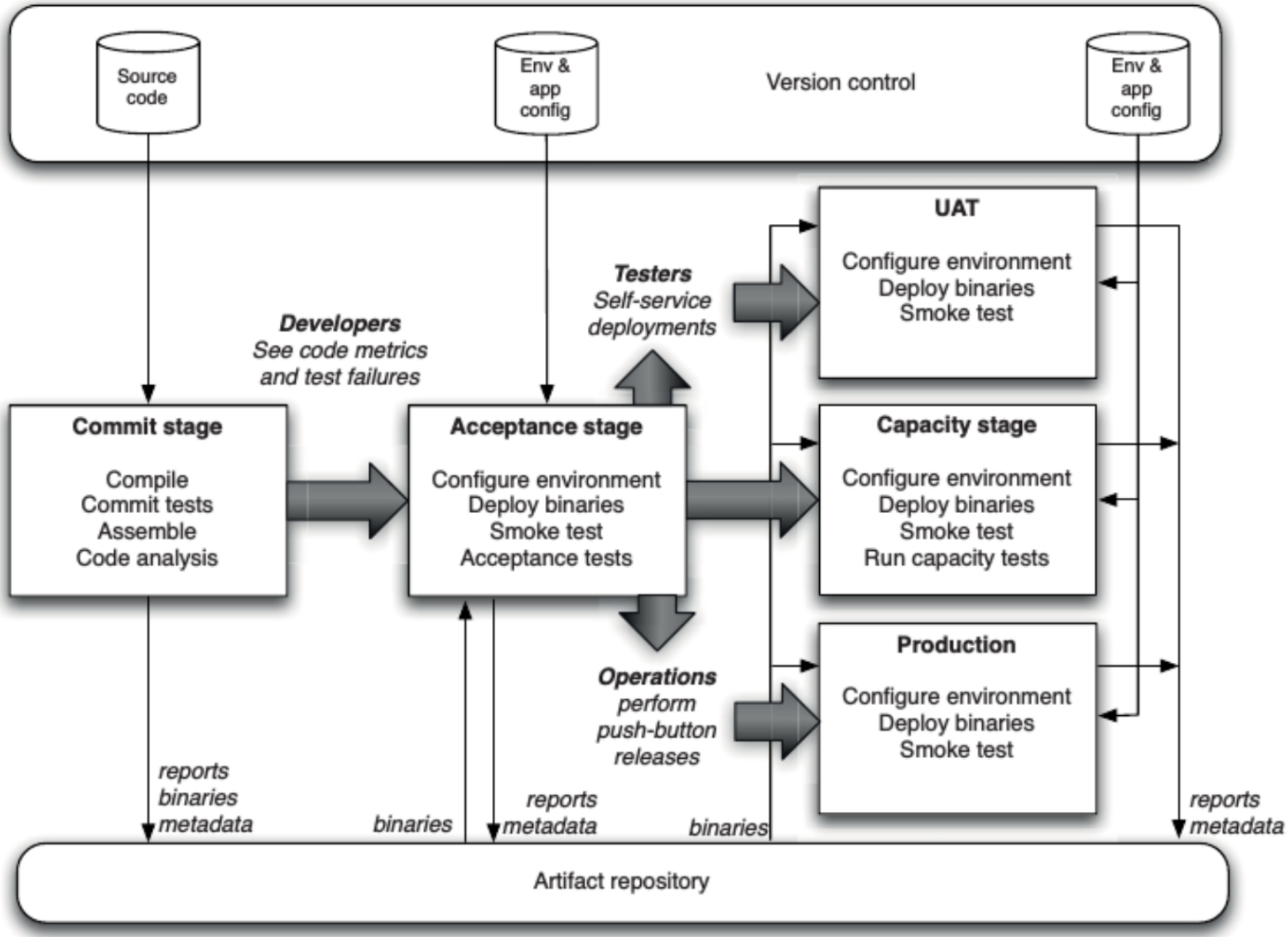
Continuous Integration: полезные практики

- Не коммитим в сломанный билд
- Проверяем все тесты локально перед коммитом
  - Проверяем мердж мастера в локальную ветку (если работаем с веткой)
  - pre-tested commits
- Ждём результата сборки на сервере перед новой задачей
- Сломанному билду нет оправданий
  - Кто сломал, тот и виноват
  - Всегда будь готов откатить изменения
  - Временной лимит на починку билда
- Настраиваем систему оповещений

Конвейер сборки



Конвейер сборки



Ещё полезные практики

- Собираем бинарники только один раз
  - Кэширование запусков конвейера
- Один и тот же процесс развёртывания для разных окружений
  - Отделение кода от конфигурации окружения
- Запуск smoke test’ов после развёртывания
- Развёртывание в копии production окружения

Ещё полезные практики

1. Проектирование основных этапов и создание скелета
2. Автоматизирование сборки и развёртывания
3. Автоматизирование модульных тестов и анализа кода
4. Автоматизирование приёмочных тестов
5. Эволюционирование конвейера

Модель зрелости процесса управления релизами

Practice	Build management and continuous integration	Environments and deployment	Release management and compliance	Testing	Data management	Configuration management
Level 3 - Optimizing: Focus on process improvement	Teams regularly meet to discuss integration problems and resolve them with automation, faster feedback, and better visibility.	All environments managed effectively. Provisioning fully automated. Virtualization used if applicable.	Operations and delivery teams regularly collaborate to manage risks and reduce cycle time.	Production rollbacks rare. Defects found and fixed immediately.	Release to release feedback loop of database performance and deployment process.	Regular validation that CM policy supports effective collaboration, rapid development, and auditable change management processes.
Level 2 - Quantitatively managed: Process measured and controlled	Build metrics gathered, made visible, and acted on. Builds are not left broken.	Orchestrated deployments managed. Release and rollback processes tested.	Environment and application health monitored and proactively managed. Cycle time monitored.	Quality metrics and trends tracked. Non functional requirements defined and measured.	Database upgrades and rollbacks tested with every deployment. Database performance monitored and optimized.	Developers check in to mainline at least once a day. Branching only used for releases.
Level 1 - Consistent: Automated processes applied across whole application lifecycle	Automated build and test cycle every time a change is committed. Dependencies managed. Re-use of scripts and tools.	Fully automated, self-service push-button process for deploying software. Same process to deploy to every environment.	Change management and approvals processes defined and enforced. Regulatory and compliance conditions met.	Automated unit and acceptance tests, the latter written with testers. Testing part of development process.	Database changes performed automatically as part of deployment process.	Libraries and dependencies managed. Version control usage policies determined by change management process.
Level 0 – Repeatable: Process documented and partly automated	Regular automated build and testing. Any build can be re-created from source control using automated process.	Automated deployment to some environments. Creation of new environments is cheap. All configuration externalized / versioned	Painful and infrequent, but reliable, releases. Limited traceability from requirements to release.	Automated tests written as part of story development.	Changes to databases done with automated scripts versioned with application.	Version control in use for everything required to recreate software: source code, configuration, build and deploy scripts, data migrations.
Level -1 – Regressive: processes unrepeatable, poorly controlled, and reactive	Manual processes for building software. No management of artifacts and reports.	Manual process for deploying software. Environment-specific binaries. Environments provisioned manually.	Infrequent and unreliable releases.	Manual testing after development.	Data migrations unversioned and performed manually.	Version control either not used, or check-ins happen infrequently.

Что почитать