FIFA Gameplay Automated Testing with Jenkins

Stuart Rowe
Electronic Arts
http://www.ea.com/

October 23, 2014

#jenkinsconf

Improving Pre-Flight Testing with Jenkins

Measures of Success:

- Increasing test throughput
- Developing a maintainable system
- Creating an accessible system

FIFA Gameplay Pre-Flight Testing

- On demand testing of a pending change
- Runs our entire test suite:
  - Automated smoke tests (offline and online)
  - Profiling and memory metrics
  - Functional gameplay tests
- Compares results against a daily baseline
- Increases confidence in each change

Limitations of Original System

- Build Pipeline composed of scripts running on a static set of machines
- No ability to parallelize or scale
- Unable to visualize the Build Pipeline
- Email reports were created manually

Why Jenkins?

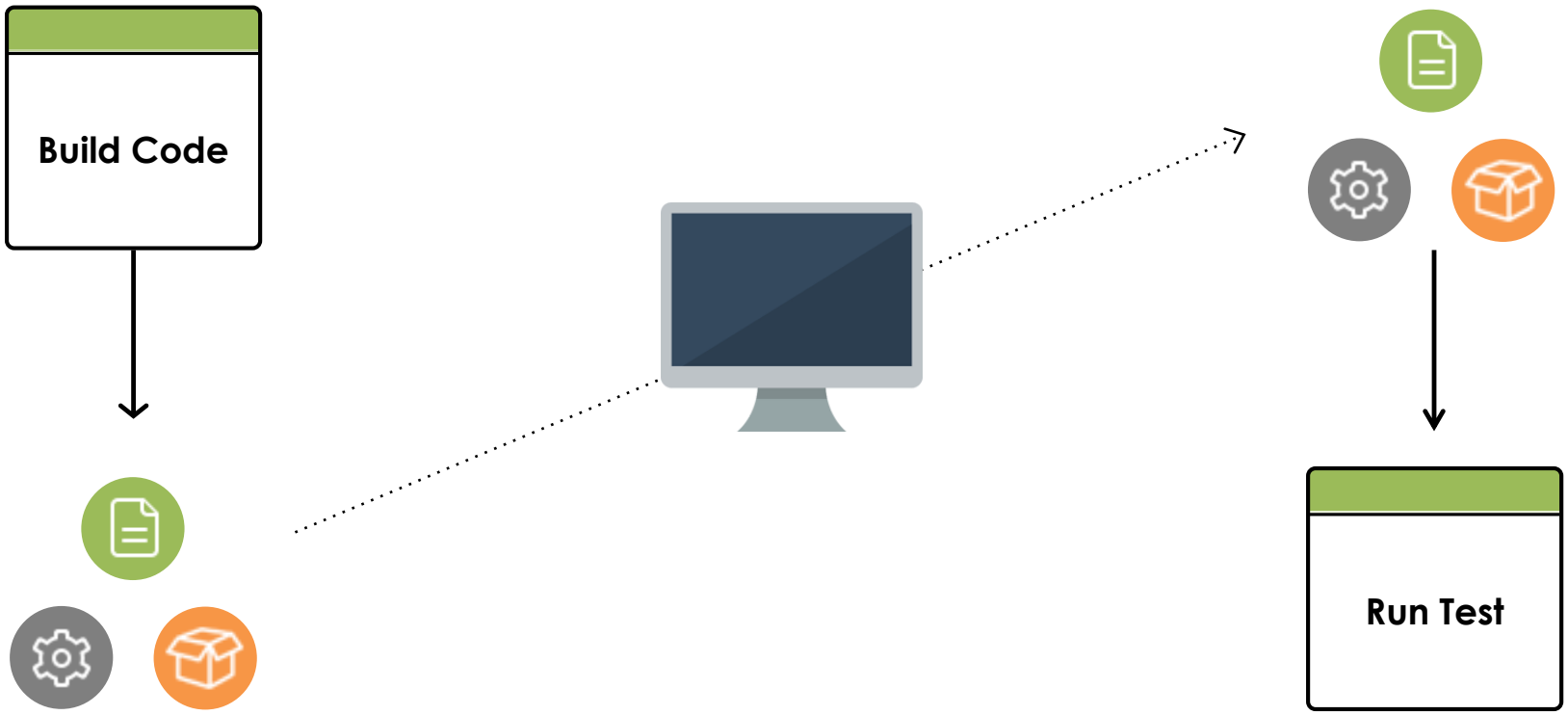
- Enables scheduling multiple tests in parallel
- Facilitates creating scalable Build Pipelines
- Provides automated reporting
- Allows Build Pipeline visualization
- Supplies accessible interface for users

Moving to Jenkins

Challenges:

- Build Archiving
- Creating Build Pipelines

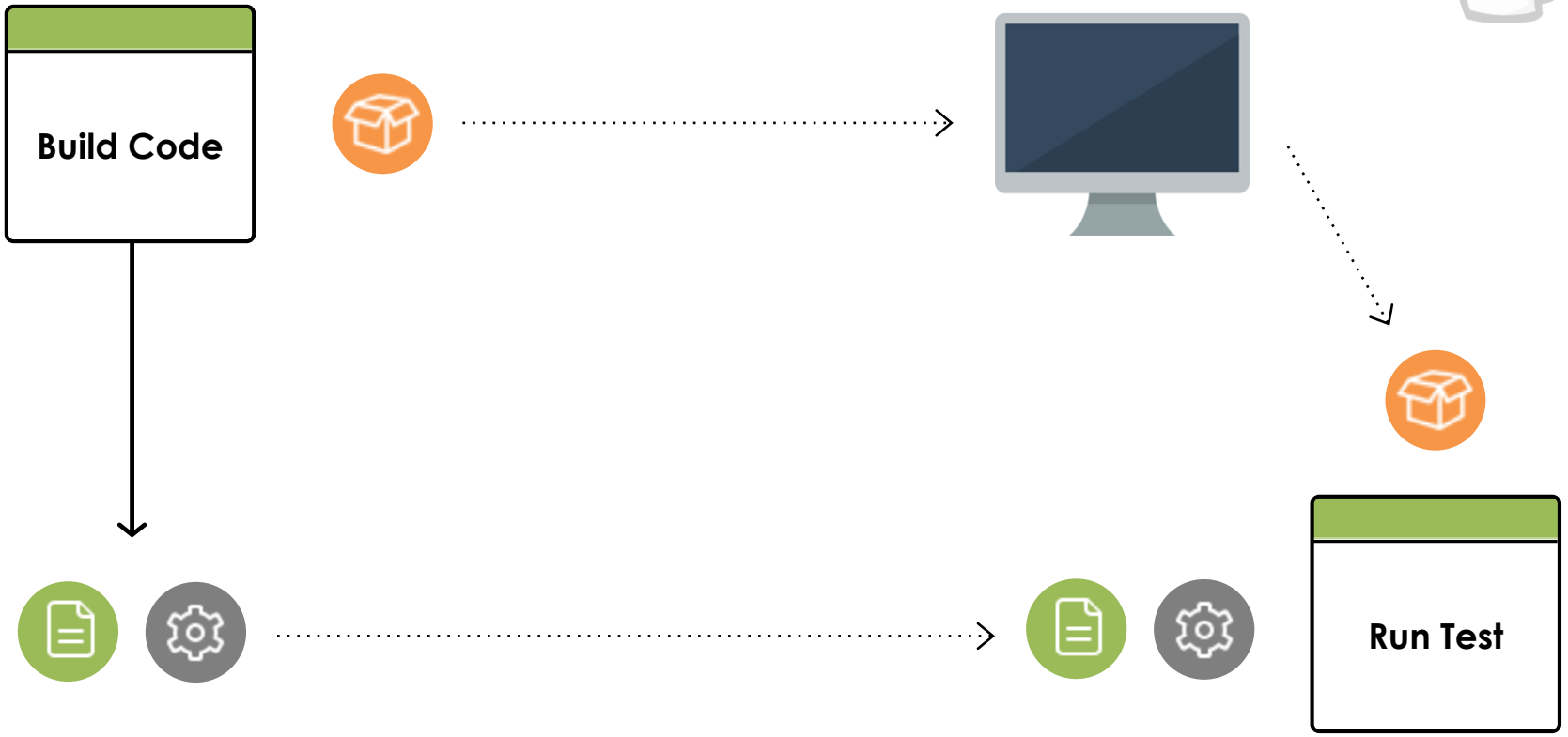
Build Archiving – Typical Pattern



Build Archiving - Challenges

- Tests run against loose builds of our game
- Build outputs include:
  - Executables, DLLs, Symbol Files, Data etc.
- Builds are on the order of 100s of GBs
- Copying all build artifacts to a central server is too expensive

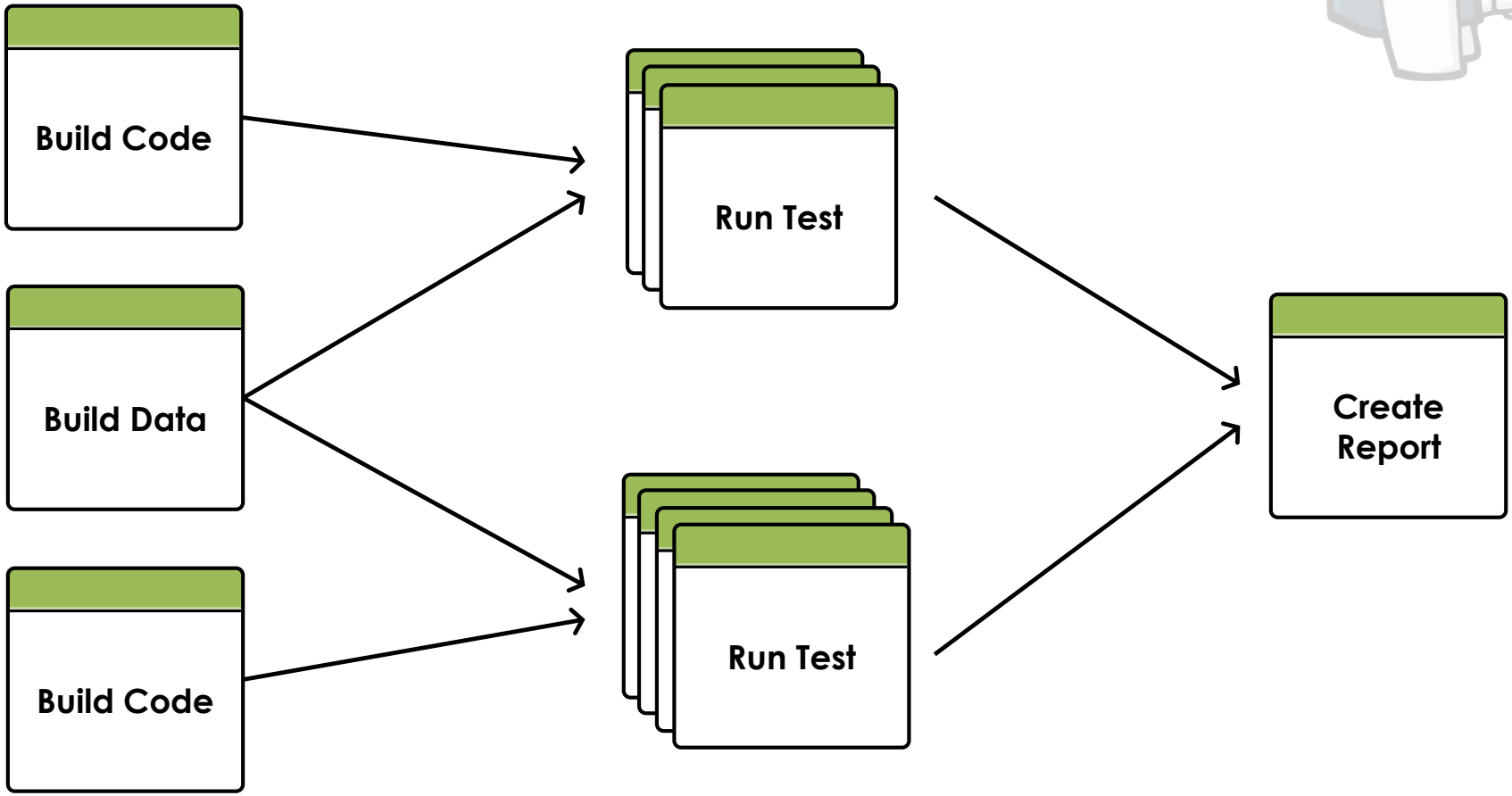
Build Archiving - Solution



Build Archiving - Solution

- Build machines each own a local build share
- Builds are copied to the shared directory
  - Only keep last n builds for each configuration
- Only small property files are copied as artifacts
  - Includes build information
- Test jobs copy small artifacts from the master
- Test scripts retrieve actual builds from the network

Build Pipelines - Design



Build Pipelines - Implementation

- Build Pipelines are defined in Build Flow projects
- Collections of closures represent Pipeline Phases
- Jobs in Phases are fundamental build steps:
  - Build code, build data, run test, submit change, etc.
- Jobs are identified by unique keys
  - Allows subsequent phases to find jobs from earlier phases
- Build Pipelines are serialized into JSON
- Reports parse JSON for Build Pipeline information

Build Pipelines - Implementation


```
// build phase
args.buildConfigs.each { config ->
    buildClosureMap[getBuildKey(config)] = generateBuildClosure(config);
}
buildJoin = parallel(buildClosureMap);

// test phase
args.testTypes.each { test ->
    args.testsConfigs[test].each { config ->
        build = buildJoin[getBuildKey(config)];
        testKey = getTestKey(config, test);
        testClosureMap[testKey] = generateTestClosure(config, test, build);
    }
}
testJoin = parallel(testClosureMap);

// report phase
reportClosure = generateReportClosure(buildJoin, testJoin, args);
build(reportClosure);
```

Project Outcomes

- Evaluated approximately 500 changes during FIFA 15 Final vs 200 changes for FIFA 14
  - Approximately 10% of changes had issues needing investigation
- More than 15,000 code builds to date
- Over 12,000 tests ran so far
- Saved 8 hours per week by automating integration tasks
- Established permanent pre-flight testing farm for engineers

Next Steps

- Move testing farm to VMs
- Improve development kit management
- Standardize process for other teams

Thank You To Our Sponsors

Platinum

CloudBees

Gold

XebiaLabs
keynote
zend
BlazeMeter
THE LOAD TESTING CLOUD ®
ravello systems
elasticbox
JFrog
perfecto mobile
black diamond SOFTWARE

Silver

SOASTA
YAHOO!
Pivotal
SERENA
choose digital
sauce LABS
opensourcearchitect

Corporate

cloudera

Thank You!

Stuart Rowe
Software Engineer, FIFA Gameplay
Electronic Arts
stuartr@ea.com