

**Macoun**

# Bring deine Tests zum Rennen

Florian Lücke (@flueckeI)

## Build result summary

### Details

Completed 23 Aug 2018, 7:01:01 PM – 1 year ago

Duration 79 minutes

[Labels](#) None 

Revision [bf66b19...](#)

Total tests 2445

Fixed in [#49](#) (Specs configuration updated)

## Build result summary

### Details

Completed 23 Aug 2018, 6:42:18 PM – 1 year ago

Duration 61 minutes

[Labels](#) None 

Revision [bf66b19...](#)

Total tests 2447

Fixed in [#59](#) (Specs configuration updated)

# Was tun wenn die Tests langsam laufen?

- Tests umbauen
- parallele Tests
  - Xcode, xctool
  - eigene Lösung

# Parallele Tests mit Xcode, xctool

# Xcode

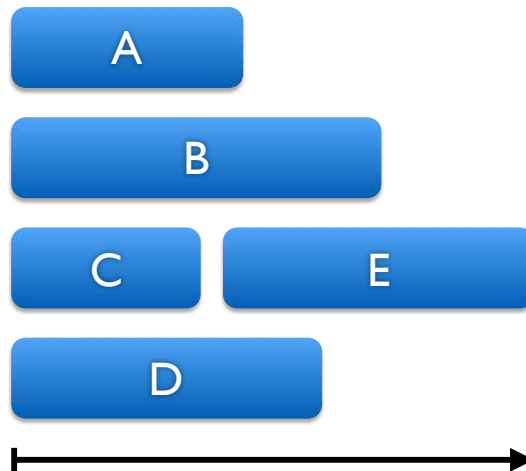
- Default:
  - Bundles, Klassen und Methoden seriell
- Seit Xcode 10:
  - Klassen eines Bundles parallel (pro Bundle einstellbar)
  - Bundles und Methoden weiter seriell

# Xcode

Xcode < 10.0



Xcode  $\geq$  10.0



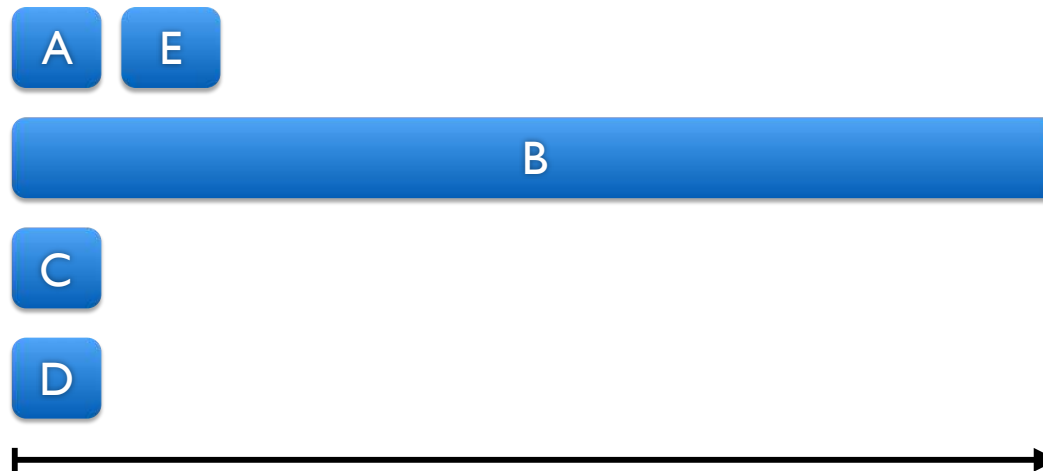


# Xcode

Xcode < 10.0



Xcode  $\geq$  10.0



# xctool

- <https://github.com/facebook/xctool>
- „At Facebook, we've seen 2x and 3x speed ups by parallelizing our runs.“
- Kann Bundles parallel ausführen
- *Möglichkeit Tests in „Buckets“ fester Größe aufzuteilen*
  - *gleicht Laufzeitunterschiede aus*

# Probleme

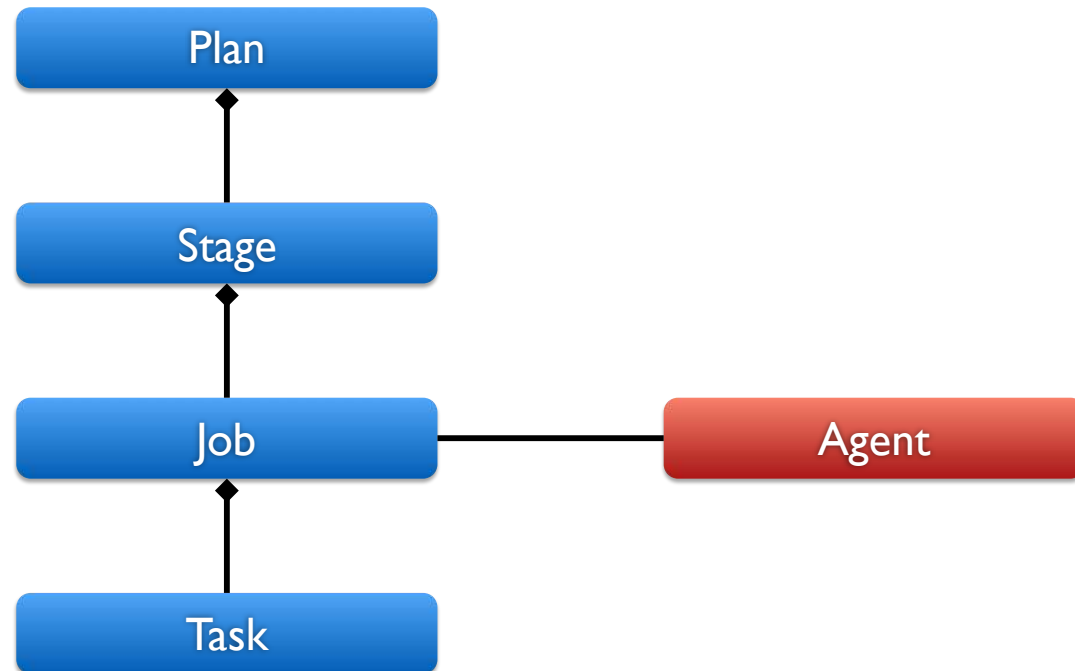
- Contested Resources
  - Race Conditions
  - Network Congestion
- Parallelität kann Tests langsamer machen oder Testergebnisse verfälschen

# Parallele Tests mit Bamboo

# Bamboo

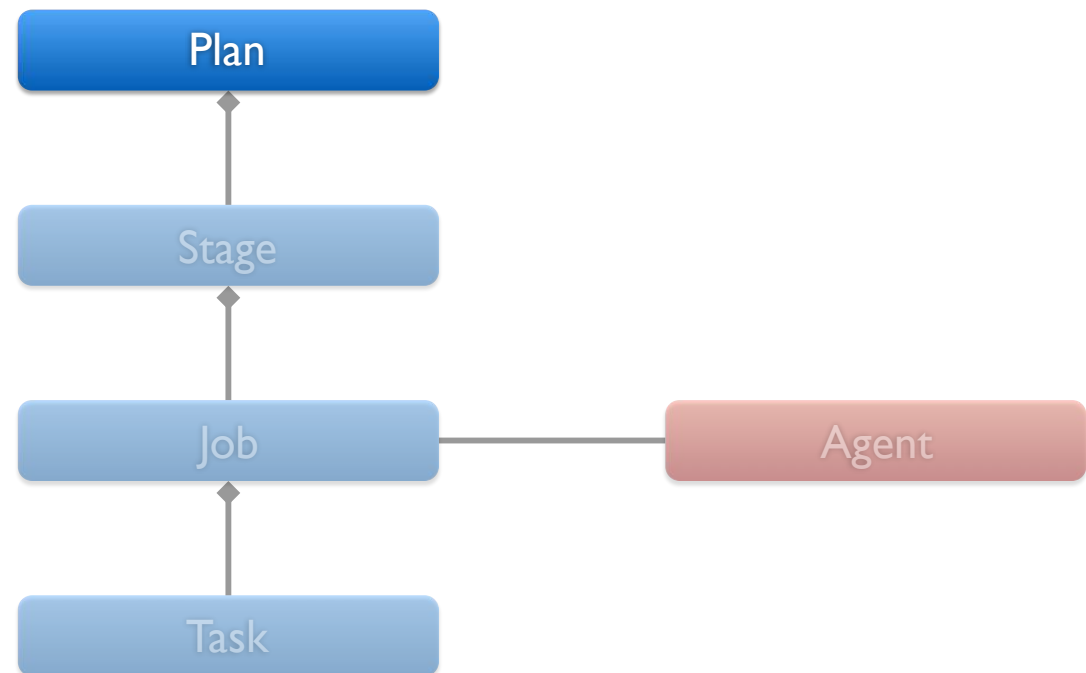
- CI/CD System von Atlassian
- Konfiguration von Plänen für verschiedene Builds
- „Agents“ führen verschiedene Aufgaben parallel aus

# Bamboo



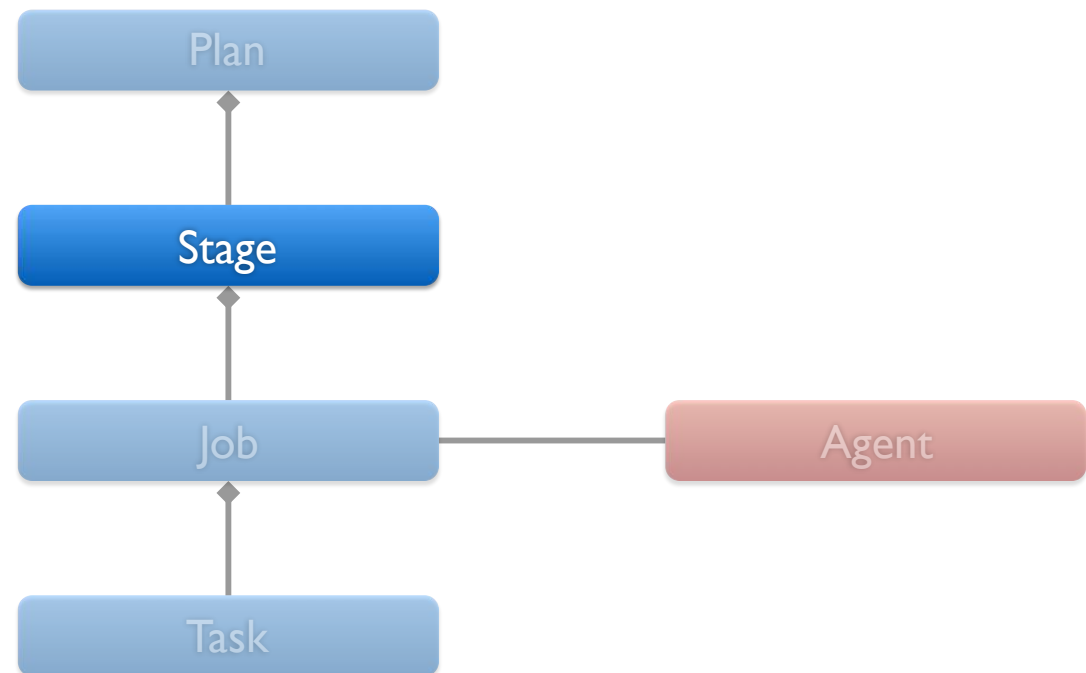
# Plans

- für verschiedene Projekte
- unabhängig ausgeführt
  - auch mehrere parallel



# Stages

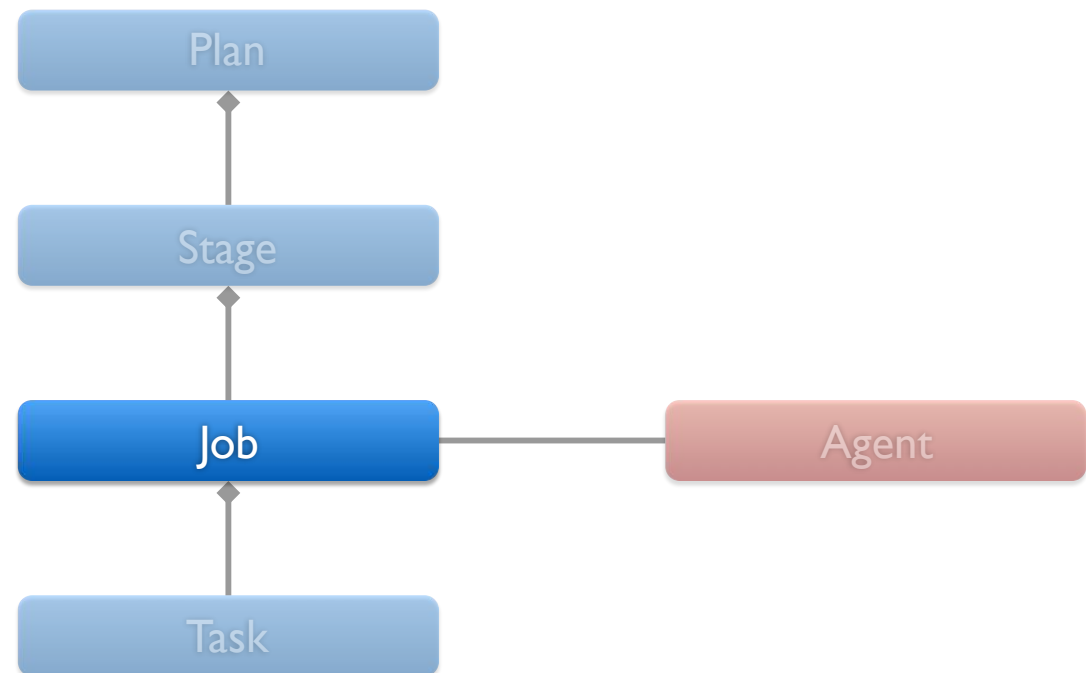
- nacheinander ausgeführt
- Stage kann Vorbedingung für die nächste sein





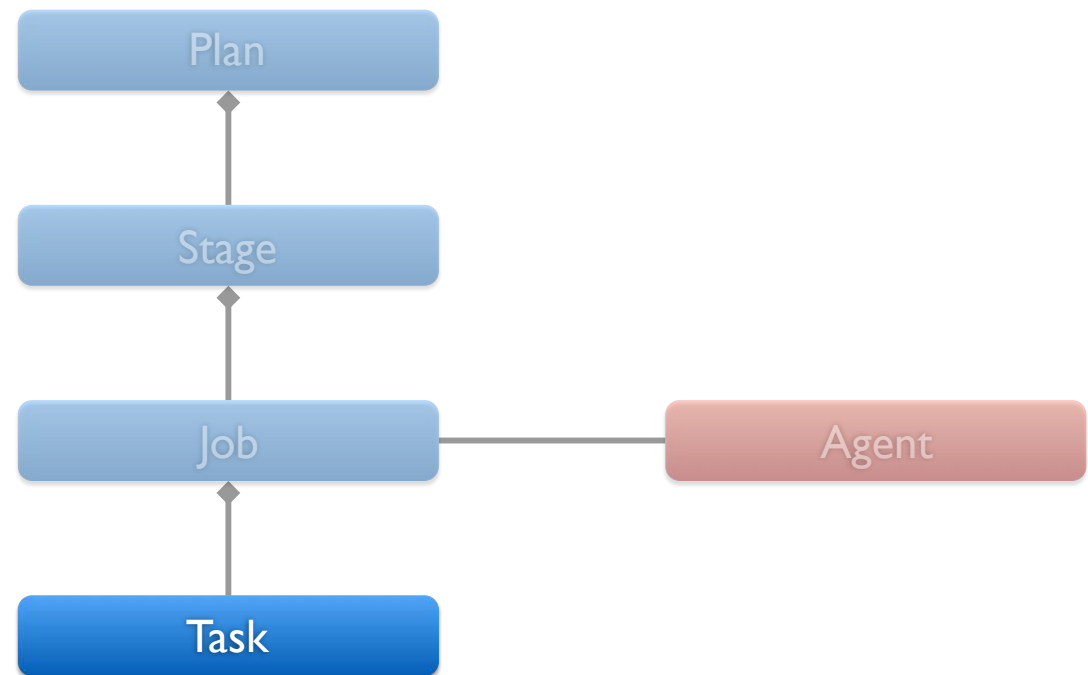
# Jobs

- parallel auf verschiedenen Agents ausgeführt
- Fehlschlag eines Jobs führt zum Fehlschlag der Stage
- beim Erreichen der Stage in Warteschleife eingereiht



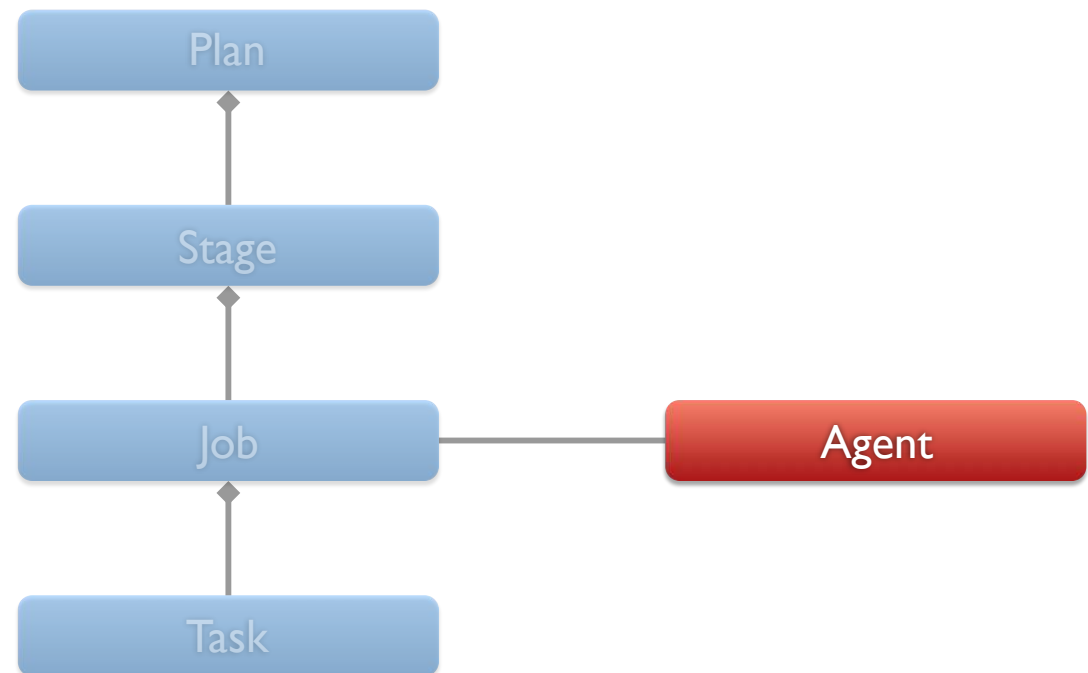
# Tasks

- nacheinander ausgeführt
- Fehlschlag eines Tasks führt zu Fehlschlag des Jobs



# Agents

- Rechner der Jobs ausführt
- remote oder lokal
- Anzahl beschränkt je nach Bamboo Lizenz



# Verteilen der Tests

- Ein Job für jedes Test Bundle
  - Bundles werden parallel auf verschiedenen Agents getestet
- Aktuell 5 Agents für parallele Verarbeitung
  - Erweiterung einfach möglich

# Artifact Sharing

- mehrfache Builds würden Testzeit erhöhen, nicht verringern
- Ideal: ein Build pro Testlauf
  - Verteilen der kompilierten Tests notwendig („Artifact Sharing“)

# Artifact Sharing

- mit Bamboo nativ möglich
  - läuft immer über Bamboo Server
  - eigene Lösung über lokales Netzwerk einfach
- xcodebuild
  - build-for-testing / test-without-building

# Upload

```
# Remove unneeded files
rm -rf "$SCHEME/ModuleCache.noindex"
rm -rf "$SCHEME/Build/Intermediates.noindex"

# Create a .tar archive containing the specified subfolders. Then upload to
# artifact server
tar -c \
  "Libraries/MarkdownKit/MarkdownKit.xcodeproj" \
  "Libraries/SyntaxKit/SyntaxKit.xcodeproj" \
  "Libraries/RTFKit/RTFKit.xcodeproj" \
  "${SCHEME}" \
  "Ulysses.xcworkspace" \
  "Ulysses.xcodeproj" | ssh ${ARTIFACT_SERVER} "cat > ${REMOTE_NAME}.tar"
```

# Upload

```
# Remove unneeded files
rm -rf "$SCHEME/ModuleCache.noindex"
rm -rf "$SCHEME/Build/Intermediates.noindex"

# Create a .tar archive containing the specified subfolders. Then upload to
# artifact server
tar -c \
  "Libraries/MarkdownKit/MarkdownKit.xcodeproj" \
  "Libraries/SyntaxKit/SyntaxKit.xcodeproj" \
  "Libraries/RTFKit/RTFKit.xcodeproj" \
  "${SCHEME}" \
  "Ulysses.xcworkspace" \
  "Ulysses.xcodeproj" | ssh ${ARTIFACT_SERVER} "cat > ${REMOTE_NAME}.tar"
```



# Upload

```
# Remove unneeded files
rm -rf "$SCHEME/ModuleCache.noindex"
rm -rf "$SCHEME/Build/Intermediates.noindex"

# Create a .tar archive containing the specified subfolders. Then upload to
# artifact server
tar -c \
  "Libraries/MarkdownKit/MarkdownKit.xcodeproj" \
  "Libraries/SyntaxKit/SyntaxKit.xcodeproj" \
  "Libraries/RTFKit/RTFKit.xcodeproj" \
  "${SCHEME}" \
  "Ulysses.xcworkspace" \
  "Ulysses.xcodeproj" | ssh ${ARTIFACT_SERVER} "cat > ${REMOTE_NAME}.tar"
```

# Download

```
# Check if the artifact has already been downloaded on the current agent
if [[ !(-d "../${ARTIFACT_NAME}") ]]; then
    scp "${ARTIFACT_SERVER_ADDRESS}:${ARTIFACT_NAME}.tar" \
        "../${ARTIFACT_NAME}/artifact.tar"
    tar -C "../${ARTIFACT_NAME}" -xf "../${ARTIFACT_NAME}/artifact.tar"
    rm "../${ARTIFACT_NAME}/artifact.tar"
fi

# Copy the cached artifact to the working directory
cp -crf "../${ARTIFACT_NAME}/." .
```

# Download

```
# Check if the artifact has already been downloaded on the current agent
if [[ !(-d "../${ARTIFACT_NAME}") ]]; then
    scp "${ARTIFACT_SERVER_ADDRESS}:${ARTIFACT_NAME}.tar" \
        "../${ARTIFACT_NAME}/artifact.tar"
    tar -C "../${ARTIFACT_NAME}" -xf "../${ARTIFACT_NAME}/artifact.tar"
    rm "../${ARTIFACT_NAME}/artifact.tar"
fi

# Copy the cached artifact to the working directory
cp -crf "../${ARTIFACT_NAME}/." .
```

# Beispiel Testlauf

1. Job der „Build“ Stage wird eingereicht

2. Job der „Build“ Stage wird ausgeführt

1. `git clone`

2. `xcodebuild build-for-testing COMPILER_INDEX_STORE_ENABLE=NO`

3. `uploadArtifacts.sh`

# Beispiel Testlauf

3. Jobs der „Test“ Stage werden eingereiht

4. Jobs der „Test“ Stage werden ausgeführt

```
1. downloadArtifacts.sh
```

```
2. xcodebuild test-without-building -only-testing "test-identifier"
```

```
3. xcresult3
```

5. Parsen und Zusammenführen der Testergebnisse von Bamboo übernommen

# Configuration As Code

- manuelle Änderungen an Tasks aufwändig und fehleranfällig
- Configuration as code: YAML-Dateien oder Java-Programm
- Versionskontrolle möglich
- Umstieg von manueller Konfiguration einfach
  - Java Spec von manuellem Plan kann generiert werden

# Configuration As Code

```
public static void main(String... argv) {  
    //By default credentials are read from the '.credentials' file.  
    BambooServer bambooServer = new BambooServer(<serverAddress>);  
  
    bambooServer.publish(iosPlan());  
    bambooServer.publish(macPlan());  
}
```

# Configuration As Code

```
public Plan iosPlan() {
    String platform = "Mobile";
    String requirementName = "system.builder.xcode.iphonesimulator13.1";
    Requirement platformRequirement = new Requirement(requirementName);
    Job[] testJobs = new Job[] {
        testJob("Controller Tests iOS", "CON", platform, platformRequirement),
        testJob("Export Tests iOS", "EXP", platform, platformRequirement),
        ...
        testJob("Utilities Tests iOS", "UTI", platform, platformRequirement)
    };
}
```



# Configuration As Code

```
final Plan iosPlan = new Plan(  
    new Project()  
        .oid(new BambooOid("1qtv4bzoleqr1"))  
        .key(new BambooKey("UL")).name("Ulysses"),  
    "iOS Tests",  
    new BambooKey("ITF")).oid(new BambooOid("1qtlf4ebdl345"));  
  
configurePlan(iosPlan, testJobs, platform, platformRequirement);  
return iosPlan;  
}
```

# Configuration As Code

```
private void plan(Plan base, Job[] testJobs, String platform, Requirement
platformRequirement) {
    base.stages(
        new Stage("Build").jobs(
            new Job("Build", new BambooKey("BUILD"))
                .tasks(new VcsCheckoutTask()
                    .description("Checkout Default Repository")
                    .checkoutItems(new CheckoutItem().defaultRepository()),
                    new ScriptTask()
                        .description("Build")
                        .inlineBody(file("build_" + platform + ".sh")),
                    new ScriptTask()
                        .description("Upload Artifacts")
                        .inlineBody(file("uploadArtifacts.sh")))
                .requirements(platformRequirement)),
            new Stage("Test").jobs(testJobs)
            ...
    )
}
```

# Configuration As Code

```
private void plan(Plan base, Job[] testJobs, String platform, Requirement
platformRequirement) {
    base.stages(
        new Stage("Build").jobs(
            new Job("Build", new BambooKey("BUILD"))
                .tasks(new VcsCheckoutTask()
                    .description("Checkout Default Repository")
                    .checkoutItems(new CheckoutItem().defaultRepository()),
                    new ScriptTask()
                        .description("Build")
                        .inlineBody(file("build_" + platform + ".sh")),
                    new ScriptTask()
                        .description("Upload Artifacts")
                        .inlineBody(file("uploadArtifacts.sh")))
                .requirements(platformRequirement)),
            new Stage("Test").jobs(testJobs))
        ...
    }
```

# Configuration As Code

```
private void plan(Plan base, Job[] testJobs, String platform, Requirement
platformRequirement) {
    base.stages(
        new Stage("Build").jobs(
            new Job("Build", new BambooKey("BUILD"))
                .tasks(new VcsCheckoutTask()
                    .description("Checkout Default Repository")
                    .checkoutItems(new CheckoutItem().defaultRepository()),
                    new ScriptTask()
                        .description("Build")
                        .inlineBody(file("build_" + platform + ".sh")),
                    new ScriptTask()
                        .description("Upload Artifacts")
                        .inlineBody(file("uploadArtifacts.sh")))
                .requirements(platformRequirement)),
            new Stage("Test").jobs(testJobs)
            ...
    )
}
```

# Configuration As Code

```
private static Job testJob(String testName, String key, String platform,
Requirement platformRequirement) {
    return new Job(testName, new BambooKey(key))
        .tasks(new ScriptTask()
            .description("Download Artifacts")
            .inlineBody(file("downloadArtifacts.sh")),
            new ScriptTask()
            .description("Run Tests")
            .inlineBody(file("test_" + platform + ".sh"))
            .argument("\"" + testName + "\"")
            new TestParserTask(TestParserTaskProperties.TestType.JUNIT)
            .resultDirectories("**/test-reports/*.xml"))
        .requirements(platformRequirement);
}
```

# Configuration As Code

```
private static Job testJob(String testName, String key, String platform,
Requirement platformRequirement) {
    return new Job(testName, new BambooKey(key))
        .tasks(new ScriptTask()
            .description("Download Artifacts")
            .inlineBody(file("downloadArtifacts.sh")),
            new ScriptTask()
            .description("Run Tests")
            .inlineBody(file("test_" + platform + ".sh"))
            .argument "\"" + testName + "\"")
            new TestParserTask(TestParserTaskProperties.TestType.JUNIT)
            .resultDirectories("**/test-reports/*.xml"))
        .requirements(platformRequirement);
}
```

# Einrichten der Agents

- eigenes Ruby Script
  - Executables werden automatisch installiert
  - nur iCloud Account anlegen und Login manuell nötig
- validiert auch Zustand der Agents

Ergebnis



# Vorher

## Build result summary

### Details

Completed 23 Aug 2018, 7:01:01 PM – 1 year ago

Duration 79 minutes

[Labels](#) None 

Revision [bf66b19...](#)

Total tests 2445

Fixed in [#49](#) (Specs configuration updated)

# Vorher

## Build result summary

### Details

Completed 23 Aug 2018, 6:42:18 PM – 1 year ago

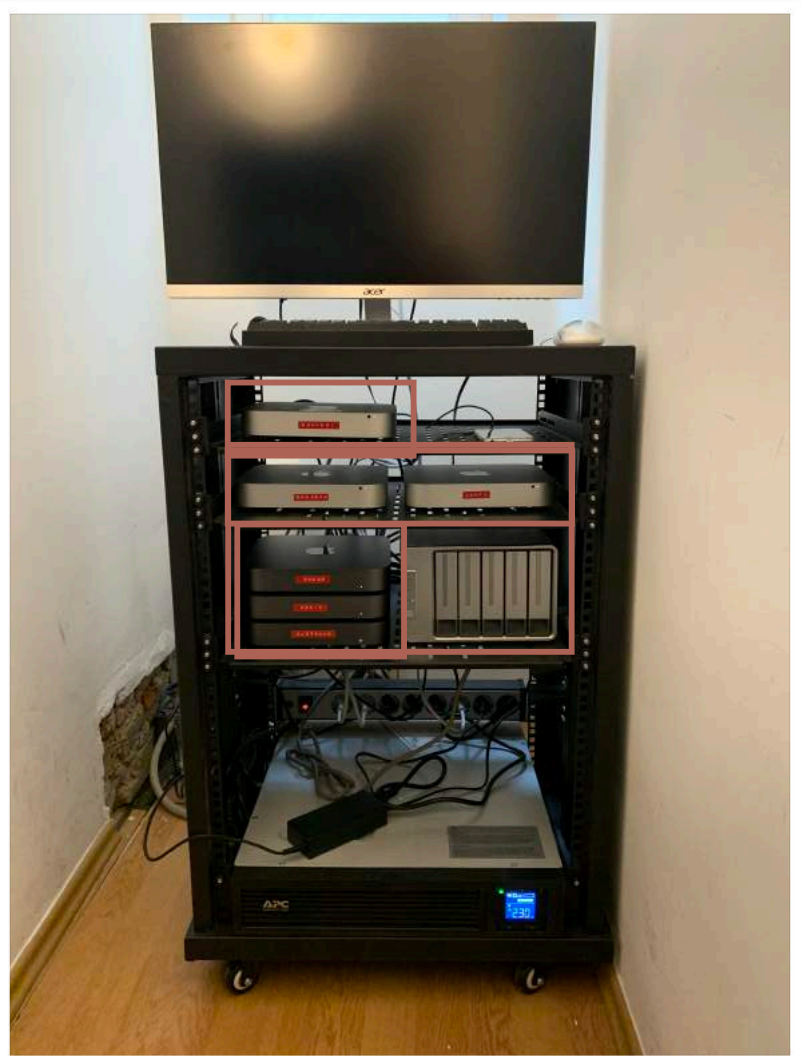
Duration 61 minutes

[Labels](#) None 

Revision [bf66b19...](#)

Total tests 2447

Fixed in [#59](#) (Specs configuration updated)



# Nachher

## Build result summary

### Details

Completed 17 Sep 2019, 11:11:19 AM – 1 hour ago

Duration 22 minutes

[Labels](#) None 

Revision [3566918...](#)

Total tests 3436

Failing since [#529](#) (Manual run by [Florian Lücke](#) – 5 days before)

# Nachher

## Build result summary

### Details

Completed 17 Sep 2019, 11:01:38 AM – 1 hour ago

Duration 21 minutes

[Labels](#) None 

Revision [3566918...](#)

Total tests 3571

Failing since [#544](#) (Changes by [Florian Lücke](#) – 5 days before)

# Probleme und Verbesserungsmöglichkeiten

- Parsen der Testergebnisse
- Plans werden bei neuen Bundles nicht automatisch aktualisiert
- weniger effektiv bei hoher Auslastung
- zusätzlich parallele Tests von Xcode verwenden
- mehrere Bamboo Agents pro Server

Fragen?

**Vielen Dank**