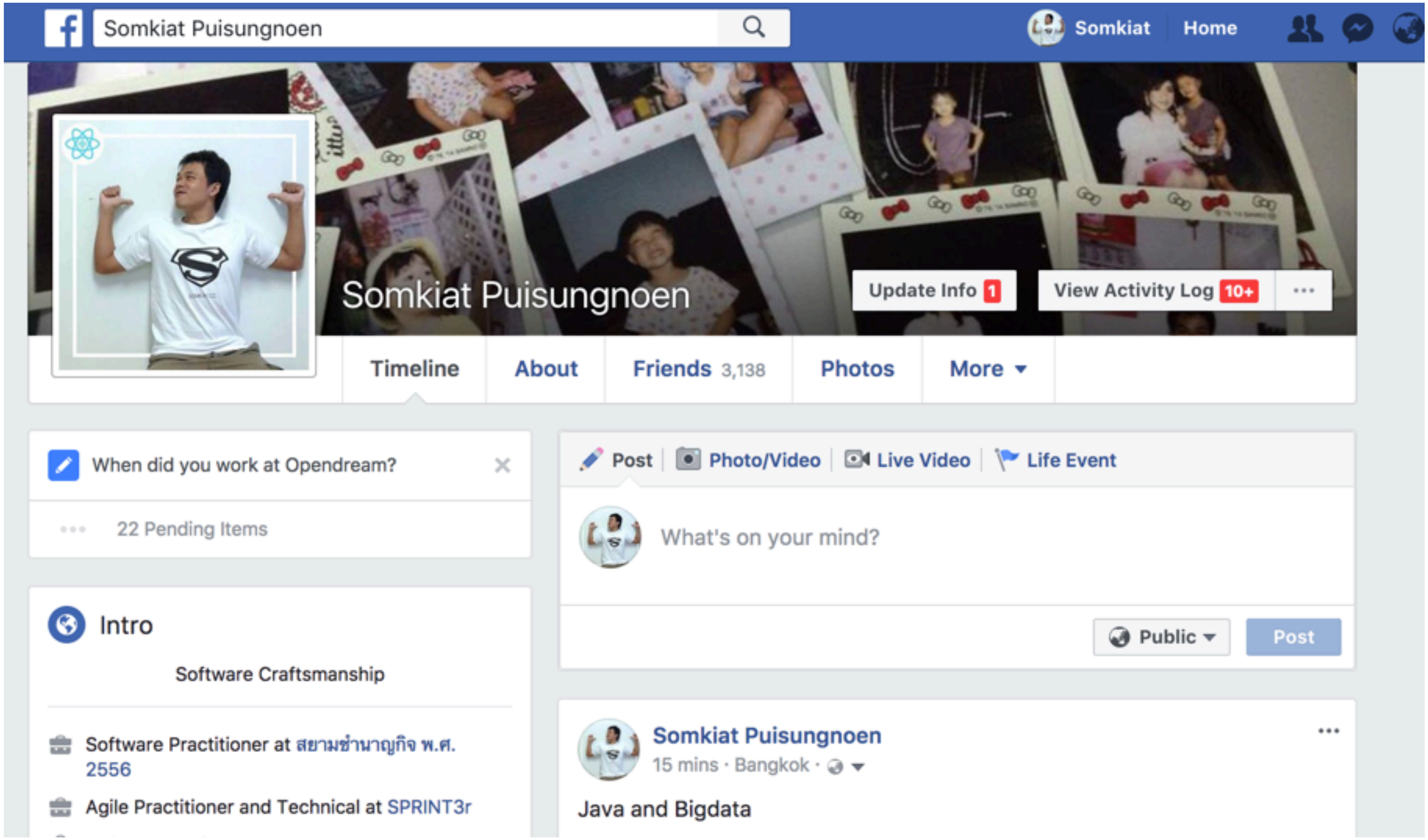
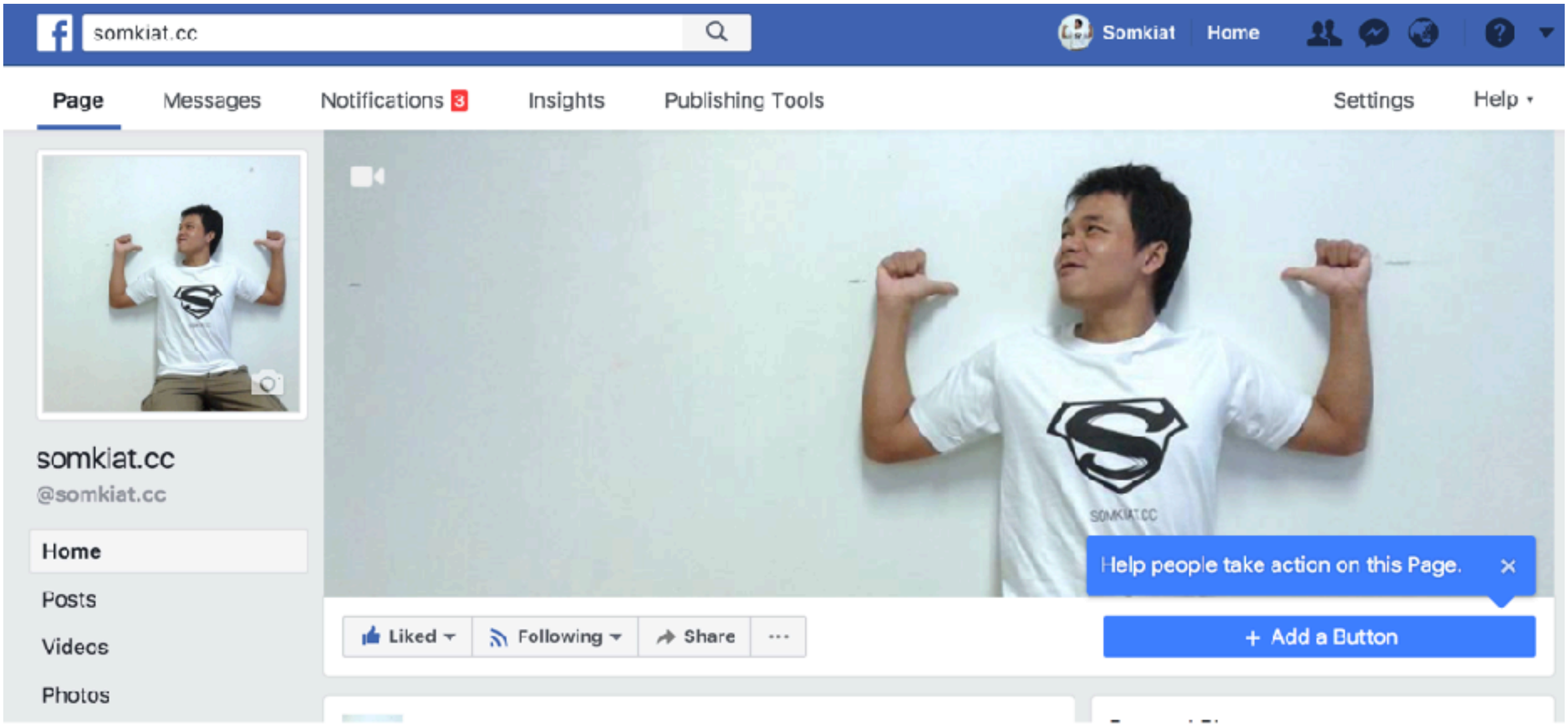
TDD series

Workshop :: TDD with Swift





Day 1

Topics (1)

1. Pyramid testing for Mobile application
2. Design system with A-DAPT blueprint
3. Overview of Xcode tool
4. Good test structure and lifecycle

Topics (1)

5. Create Unit tests with XCTest
6. Create UI tests with XCTest
7. Good unit test with FIRST
8. Install fastland tool => scan, snapshot
9. Code coverage

Code coverage with Slather

SLATHER

Apply tests liberally

https://github.com/SlatherOrg/slather

Code coverage with Slather

$fastlane scan -s UI

$slather coverage -s --scheme UI
IMC_Calculator.xcodeproj

Day 2

Topics

1. Overview application
2. Start to write tests
3. Code coverage
4. Review
5. Build your CI/CD system with Jenkins

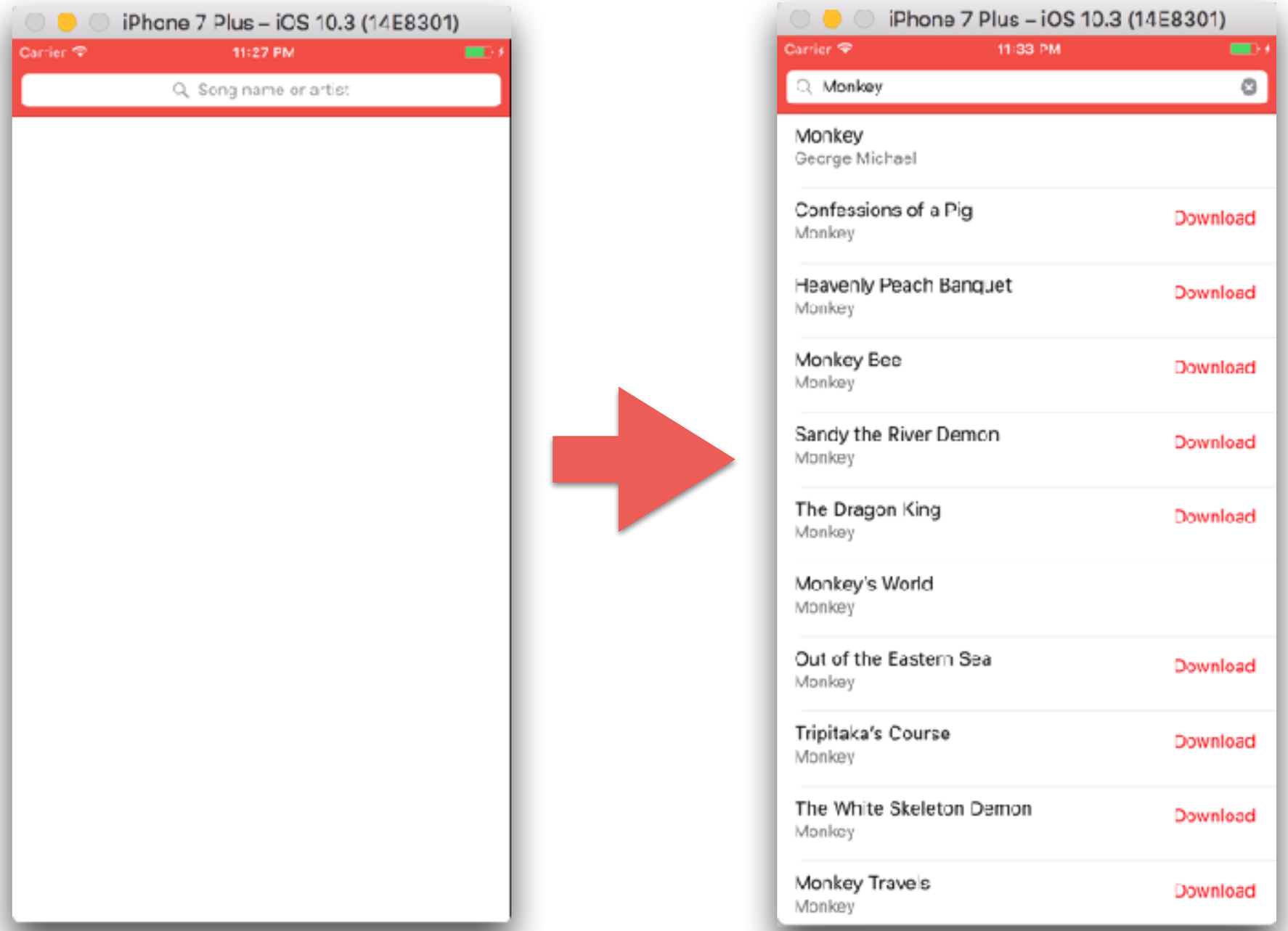
https://github.com/up1/workshop-starter-tdd-swift

Overview application

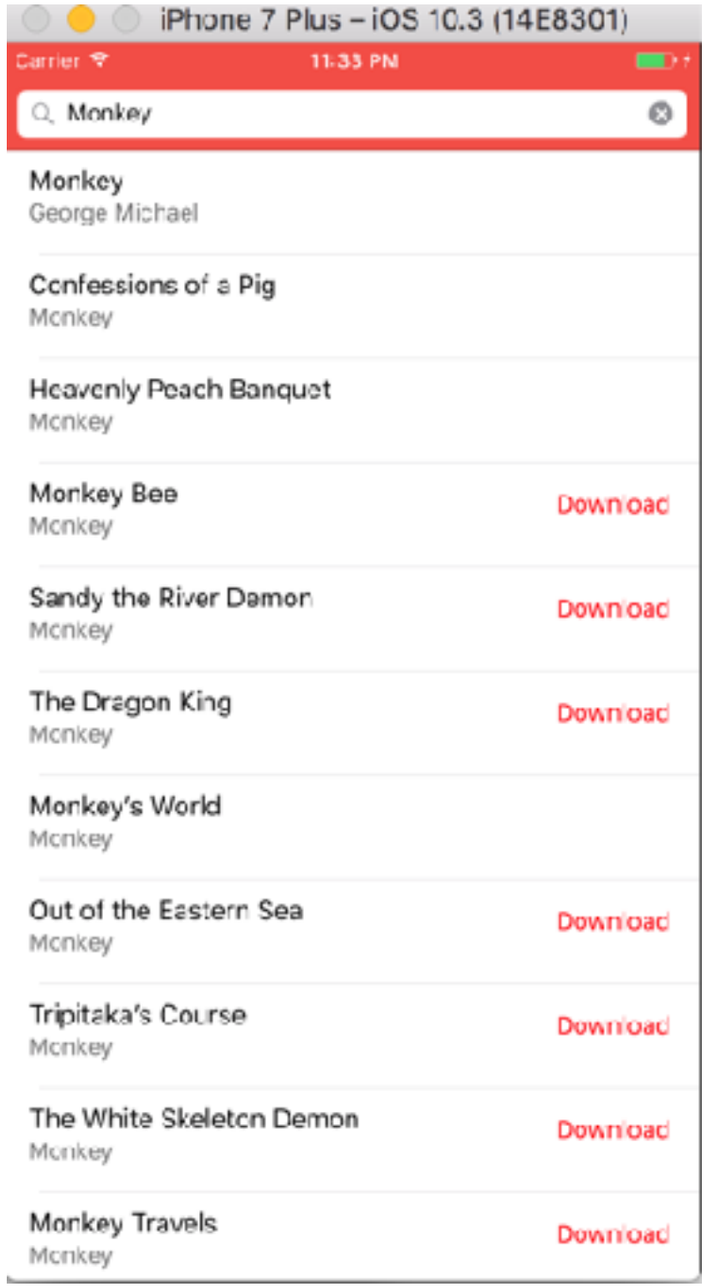
Architecture

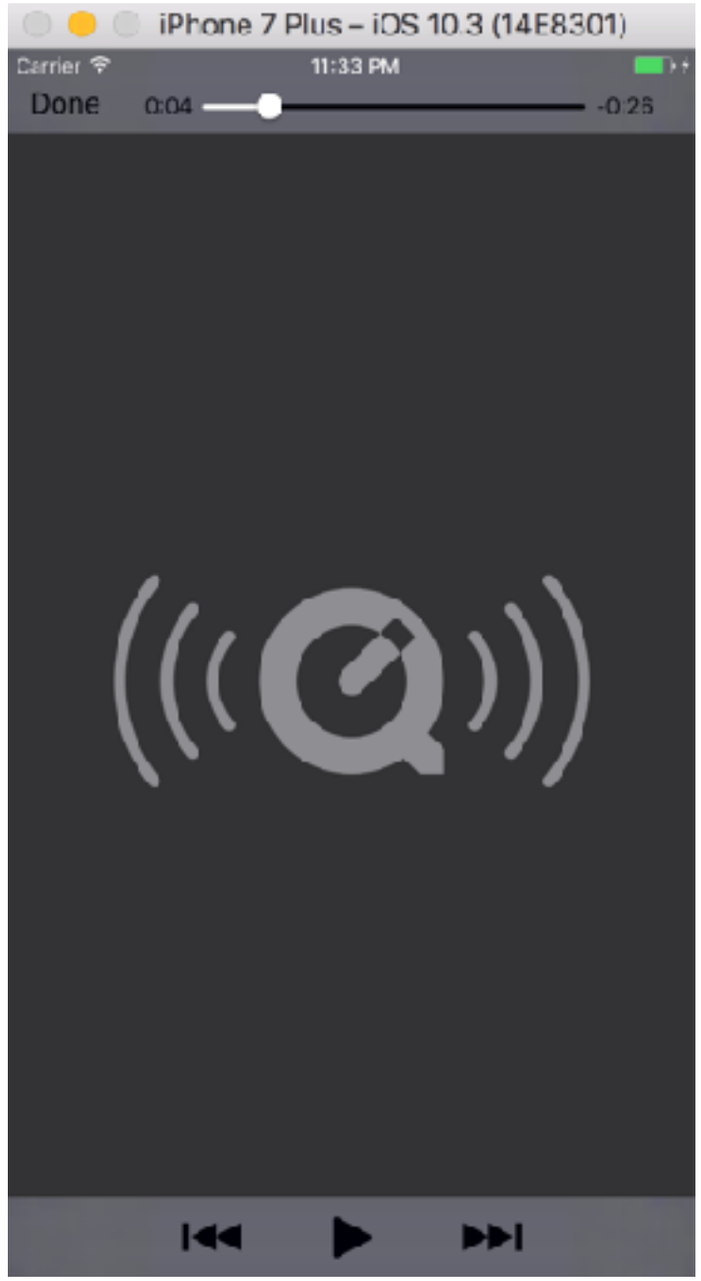
HTTP

Search songs from iTune (1)

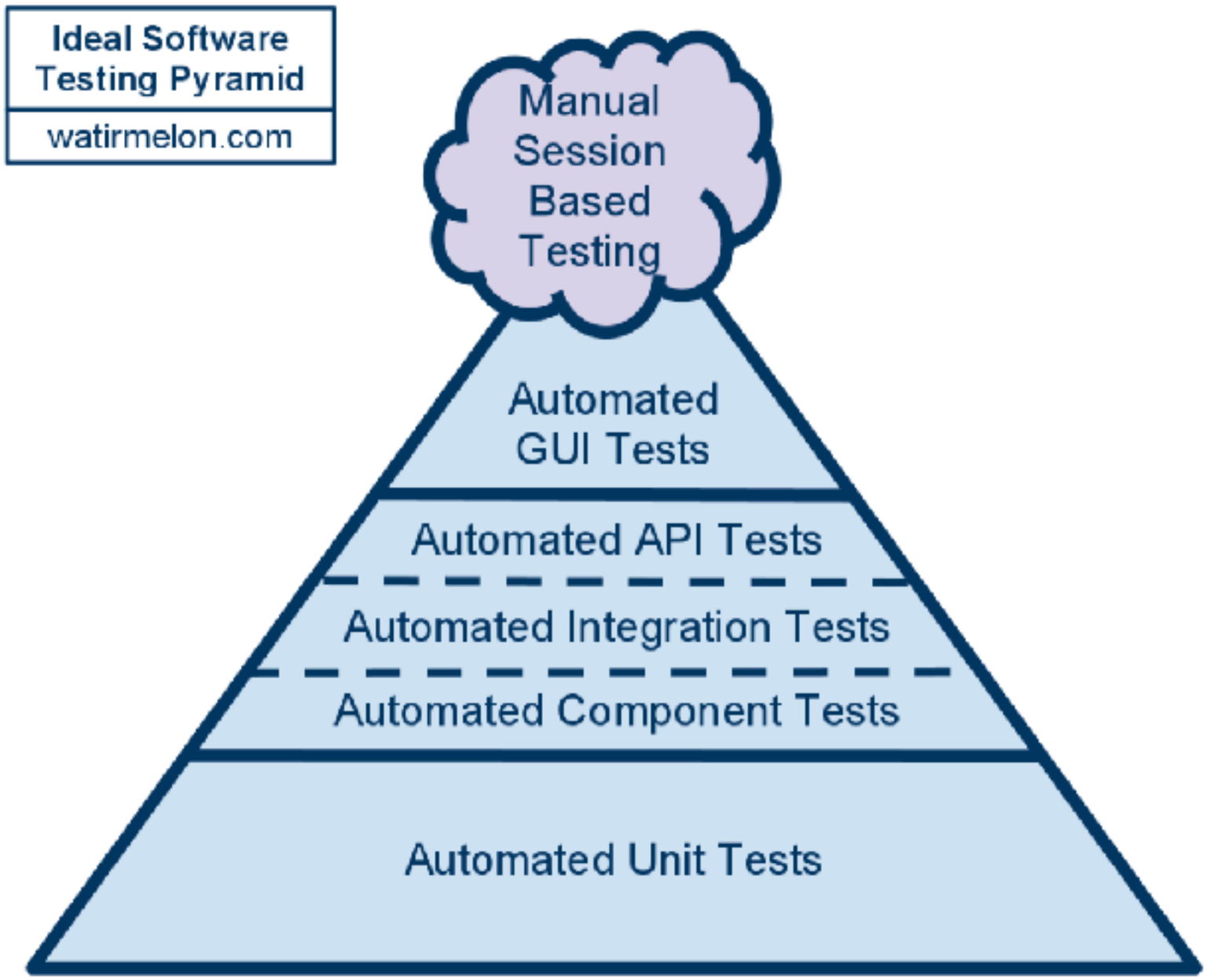


Search songs from iTune (1)

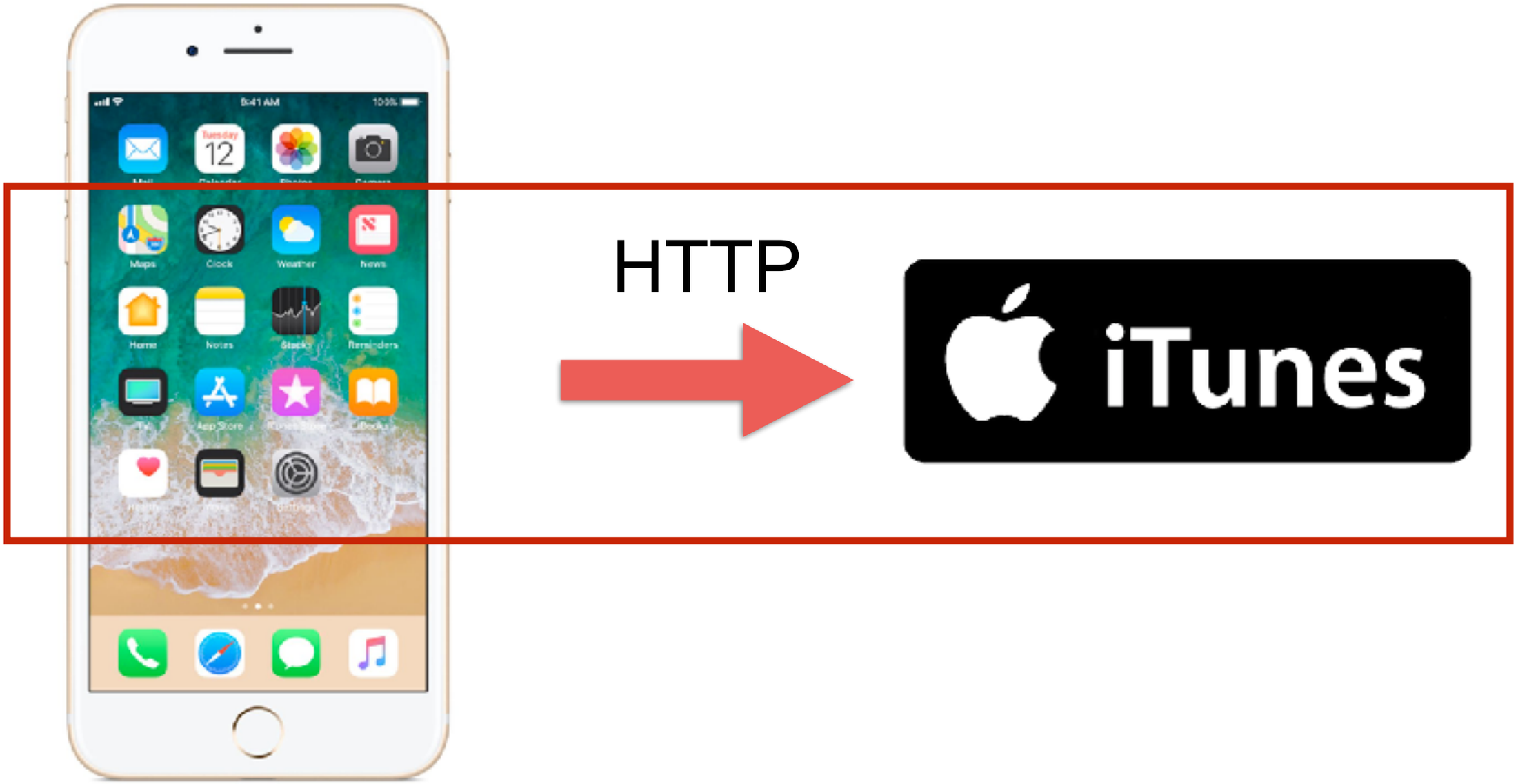


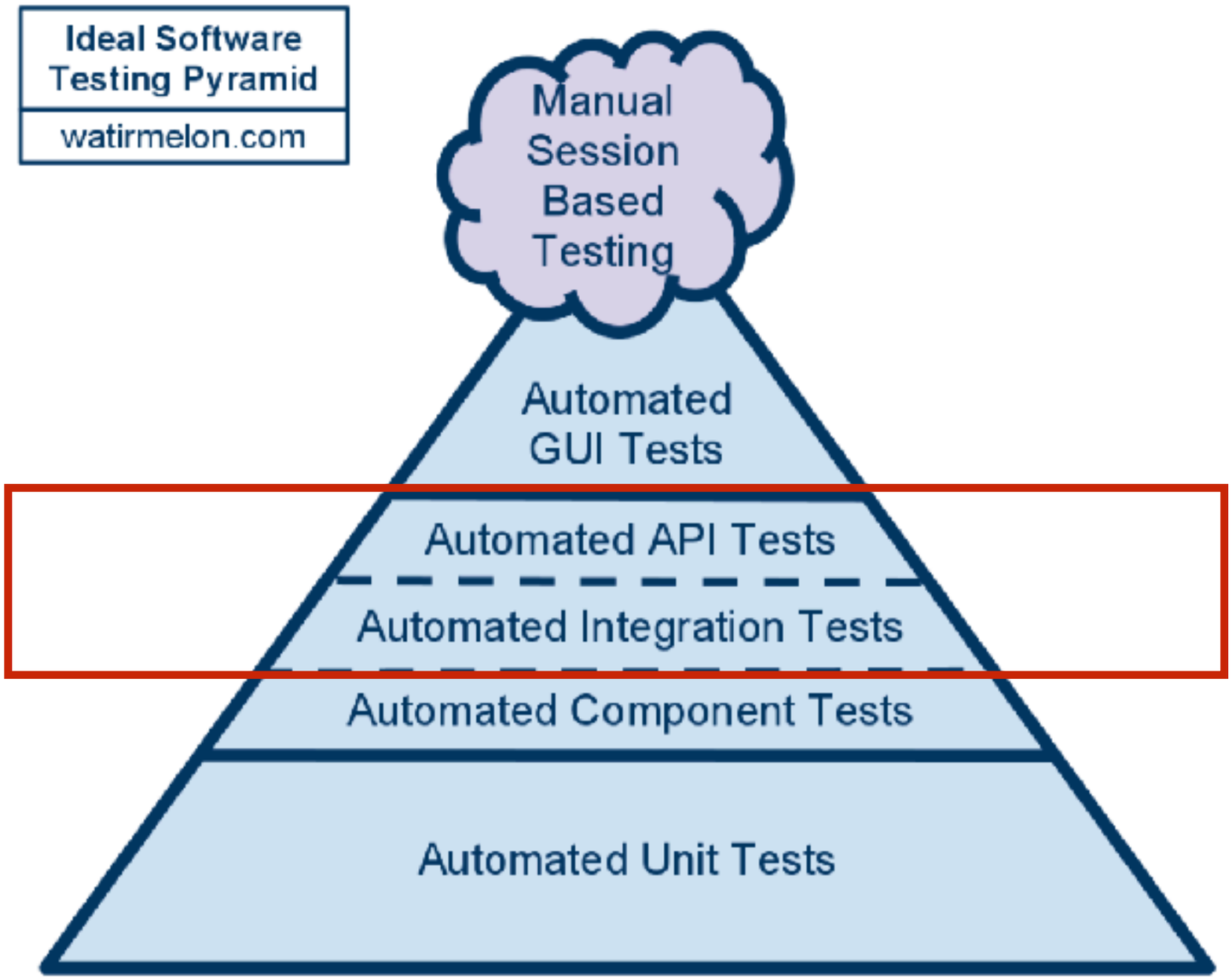


What to test ?



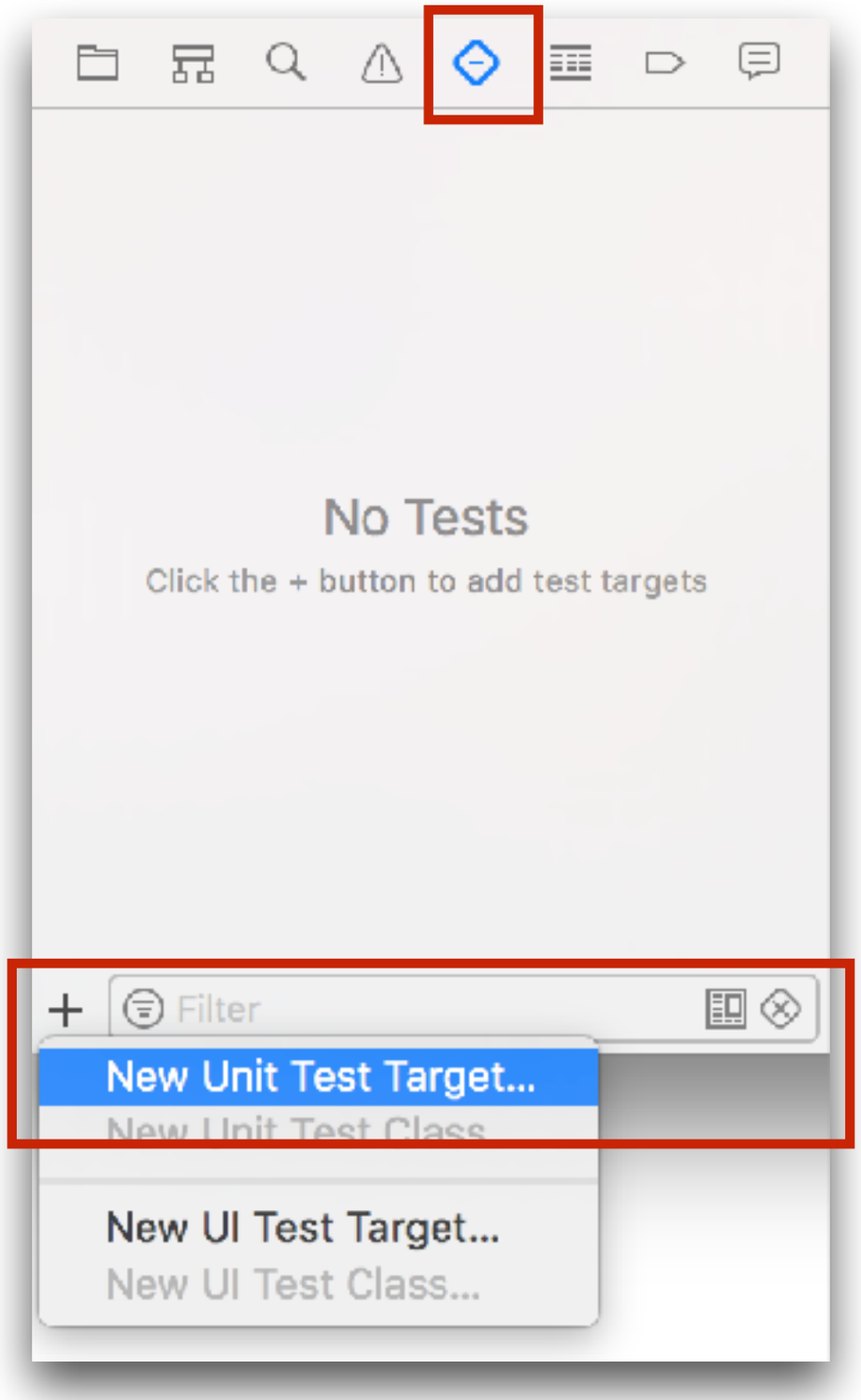
Call iTune API (Integration test)





How to test ?

Add Unit tests target to project



First test :: Call real API

1. Create new test class

```
import XCTest

@testable import HalfTunes

class HalTunesTests: XCTestCase {

    override func setUp() {
        super.setUp()
    }

    override func tearDown() {
        super.tearDown()
    }

    func testExample() {
    }

}
```

2. Session to send request

```
class HalTunesTests: XCTestCase {

    var sessionUnderTest: URLSession!

    override func setUp() {
        super.setUp()
        sessionUnderTest = URLSession(configuration:
                                      URLSessionConfiguration.default)
    }

    override func tearDown() {
        sessionUnderTest = nil
        super.tearDown()
    }

}
```

3. Start to write a test (1)

Arrange to set URL and expectation result

```
func testSuccess_Call_To_iTune_API_Should_Retuen_Status_code_200() {

        // Arrange
        let url = URL(string: "https://itunes.apple.com/search?
media=music&entity=song&term=monkey")

        // What i expect to happen
        let promise = expectation(description: "Status code: 200")


}
```

3. Start to write a test (2)

```
// Act
let dataTask = sessionUnderTest.dataTask(with: url!) { data, response, error in
    // Assert
    if let error = error {
        XCTFail("Error: \(error.localizedDescription)")
        return
    } else if let statusCode = (response as? HTTPURLResponse)?.statusCode {
        if statusCode == 200 {
            // Success with condition
            promise.fulfill()
        } else {
            XCTFail("Status code: \(statusCode)")
        }
    }
}
dataTask.resume()

// Test running util all expectations are fulfilled
waitForExpectations(timeout: 5, handler: nil)
```

Second test :: Call real API

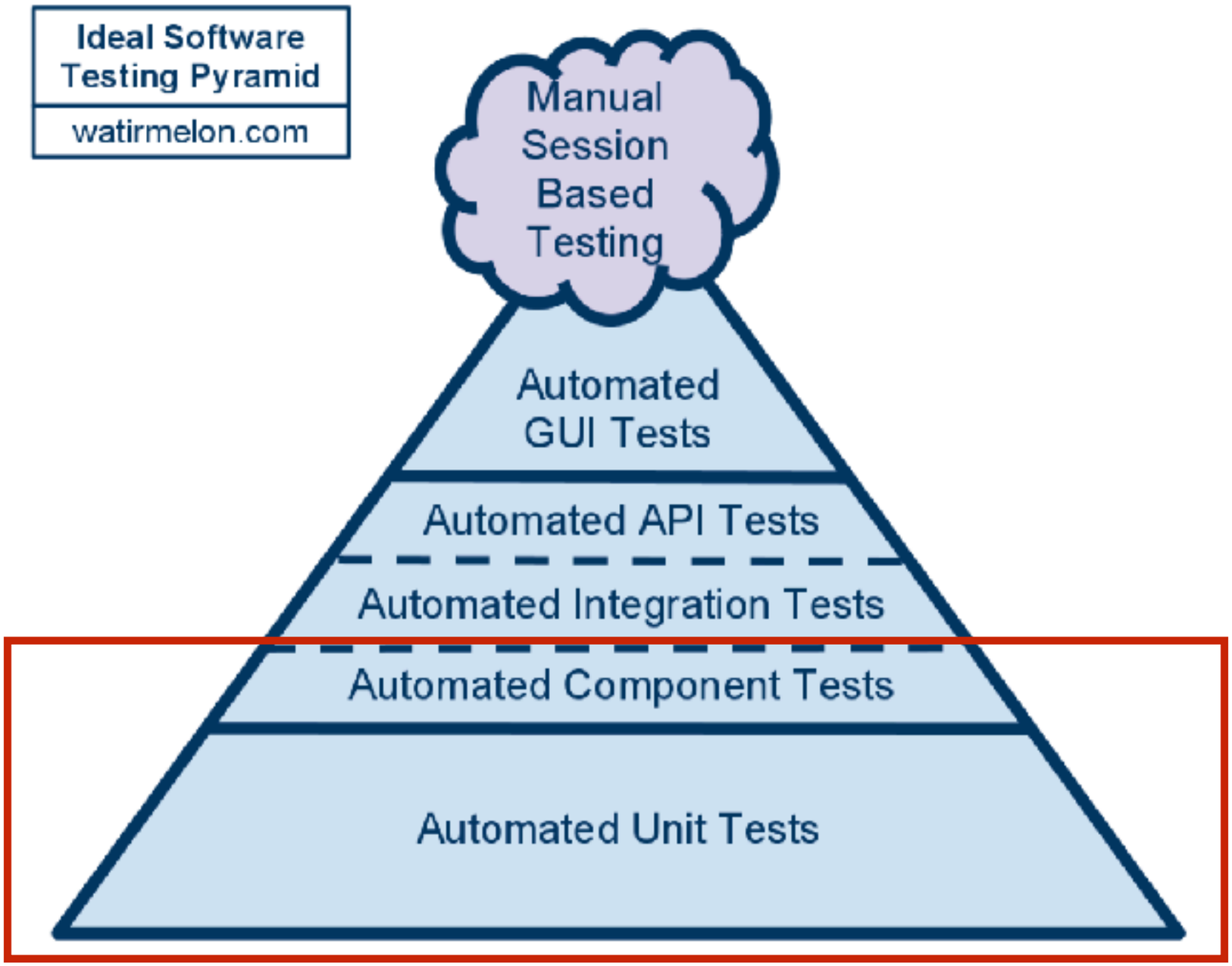
Fail Faster ?

Fail Faster

```
let promise = expectation(description: "Completion handler invoked")
var statusCode: Int?
var responseError: Error?

// Act
let dataTask = sessionUnderTest.dataTask(with: url!) {
     data, response, error in

     statusCode = (response as? HTTPURLResponse)?.statusCode
     responseError = error
     promise.fulfill()
}
dataTask.resume()

waitForExpectations(timeout: 5, handler: nil)

// Assert
XCTAssertNil(responseError)
XCTAssertEqual(statusCode, 200)
```

Call iTune API (Integration test)

What next to test ?



Unit test ?

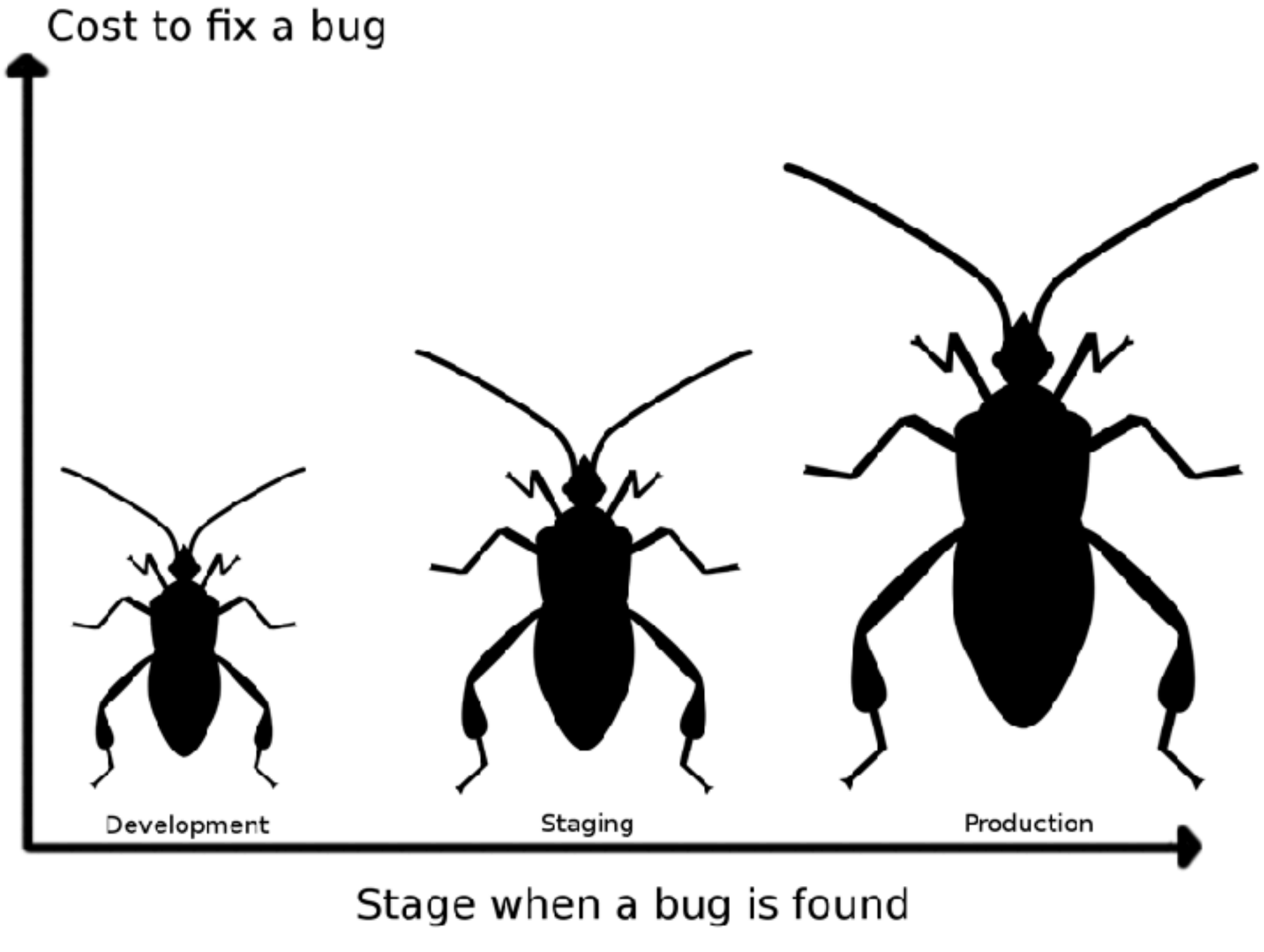
Continuous Integration

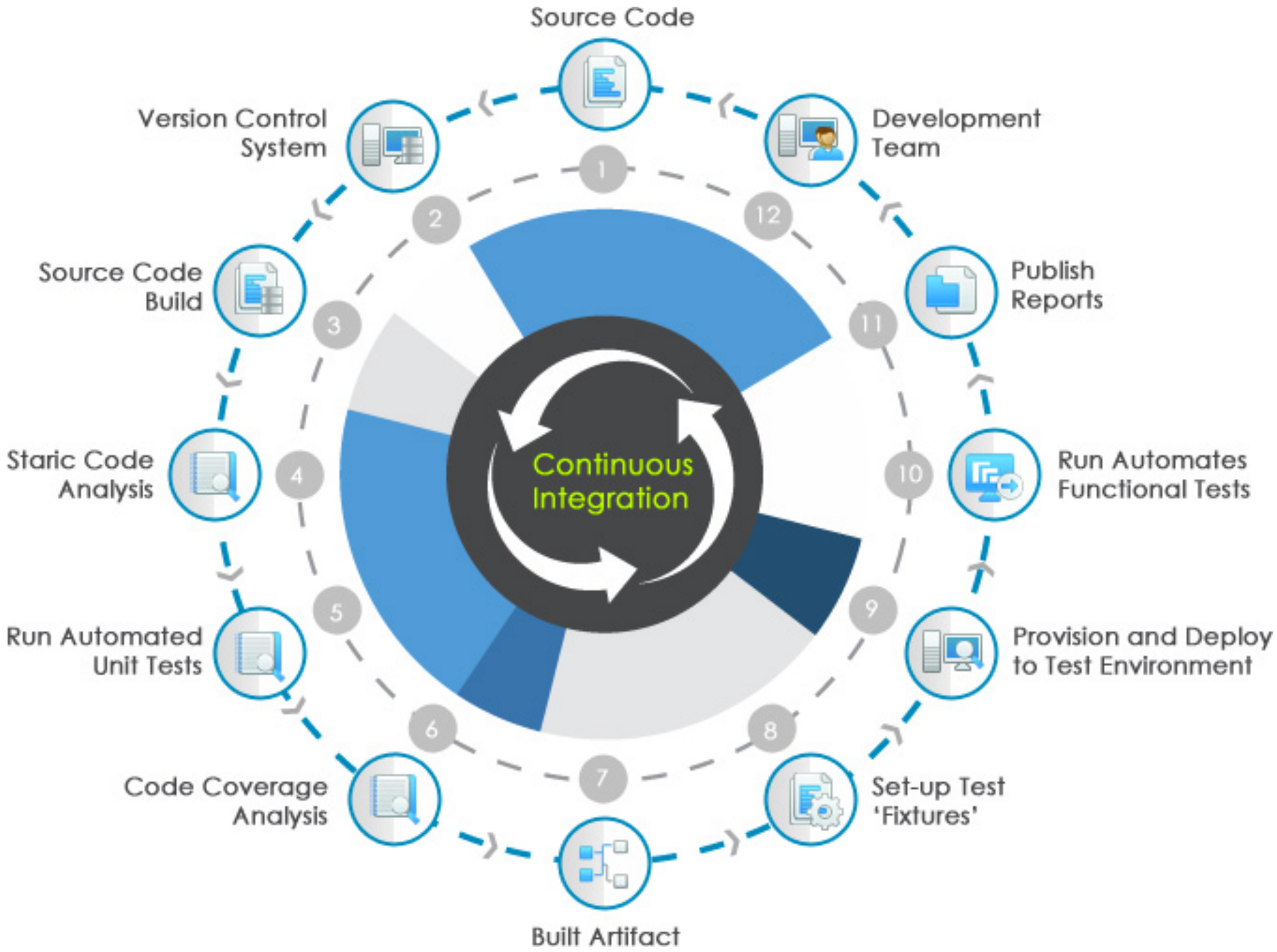
Why CI/CD ?

The cost of integration

1. Merging the code
2. Duplicate changes
3. Test again again !!
4. Fixing bugs
5. Impact on stability

The cost of integration





Jenkins

Bamboo

TeamCity

go™

Visual Studio Team Foundation Server

Hudson

Travis

wercker

circleci

CI is about what people do not about what tools they use

Continuous Integration

Discipline to **integrate frequently**

Continuous Integration

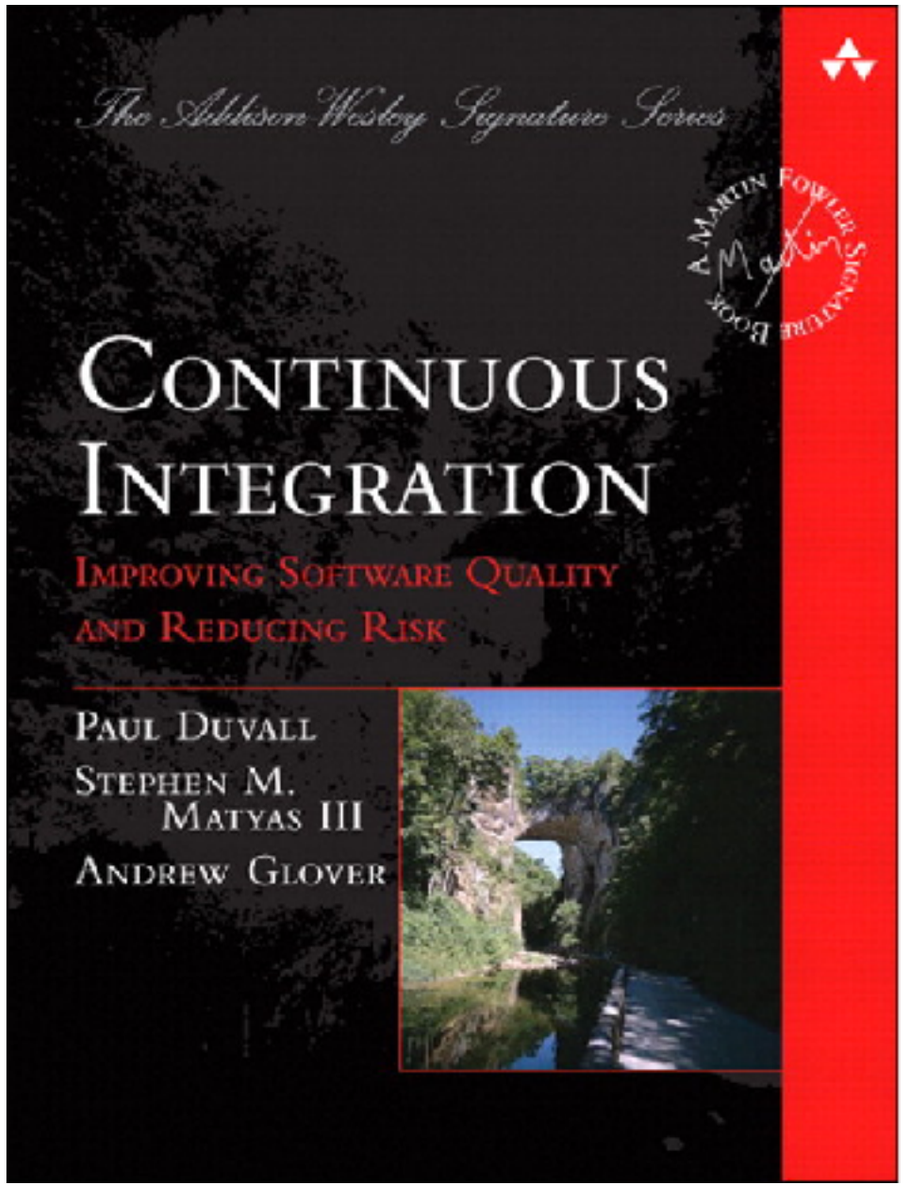
Strive to make **small change**

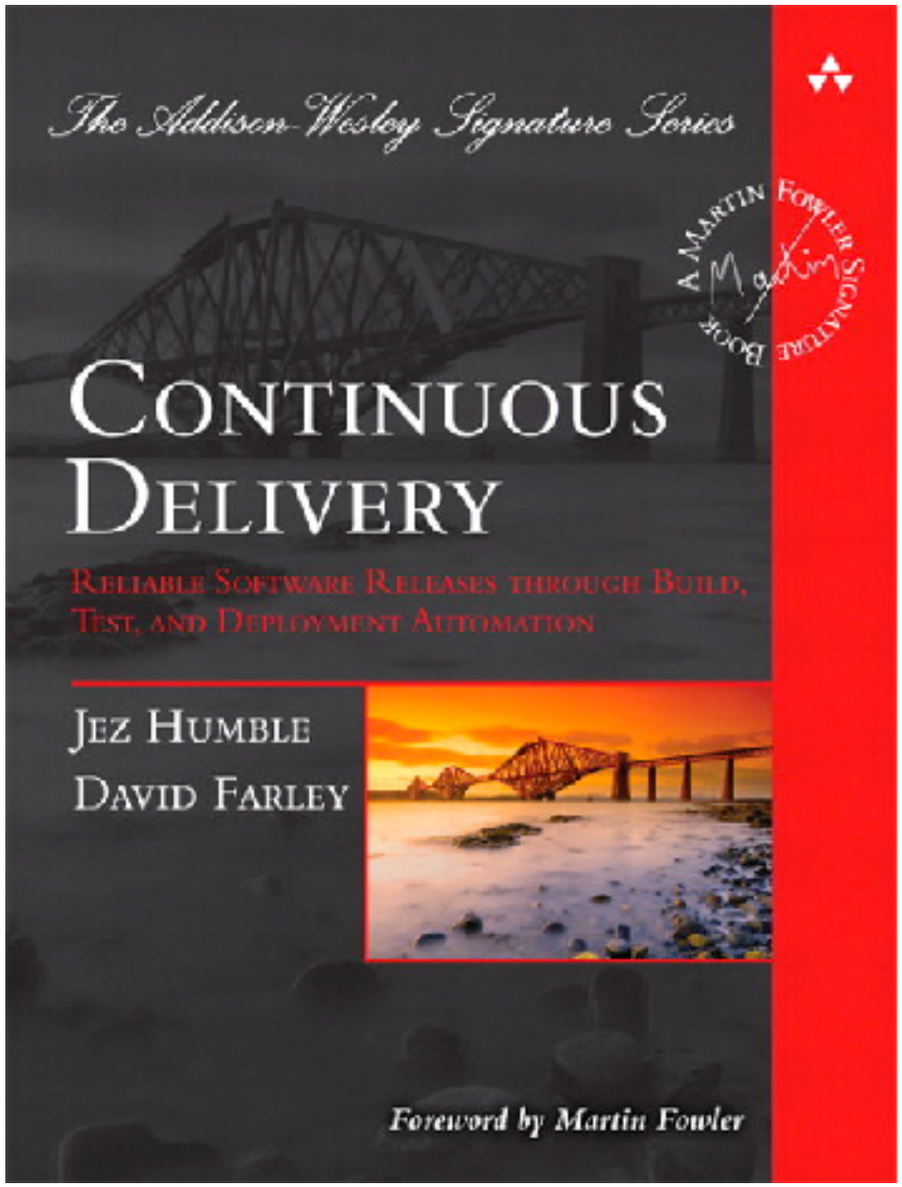
Continuous Integration

Strive for **fast feedback**

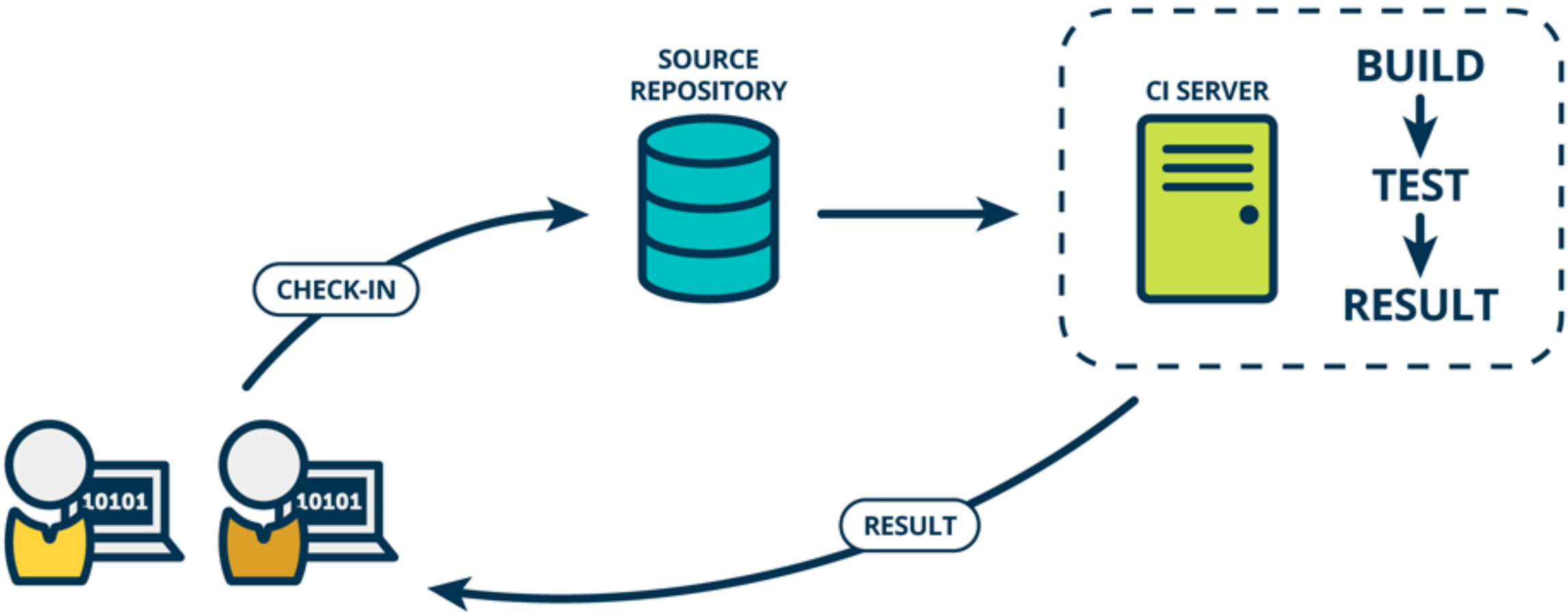
Practices of Continuous Integration

Improve quality and reduce risk

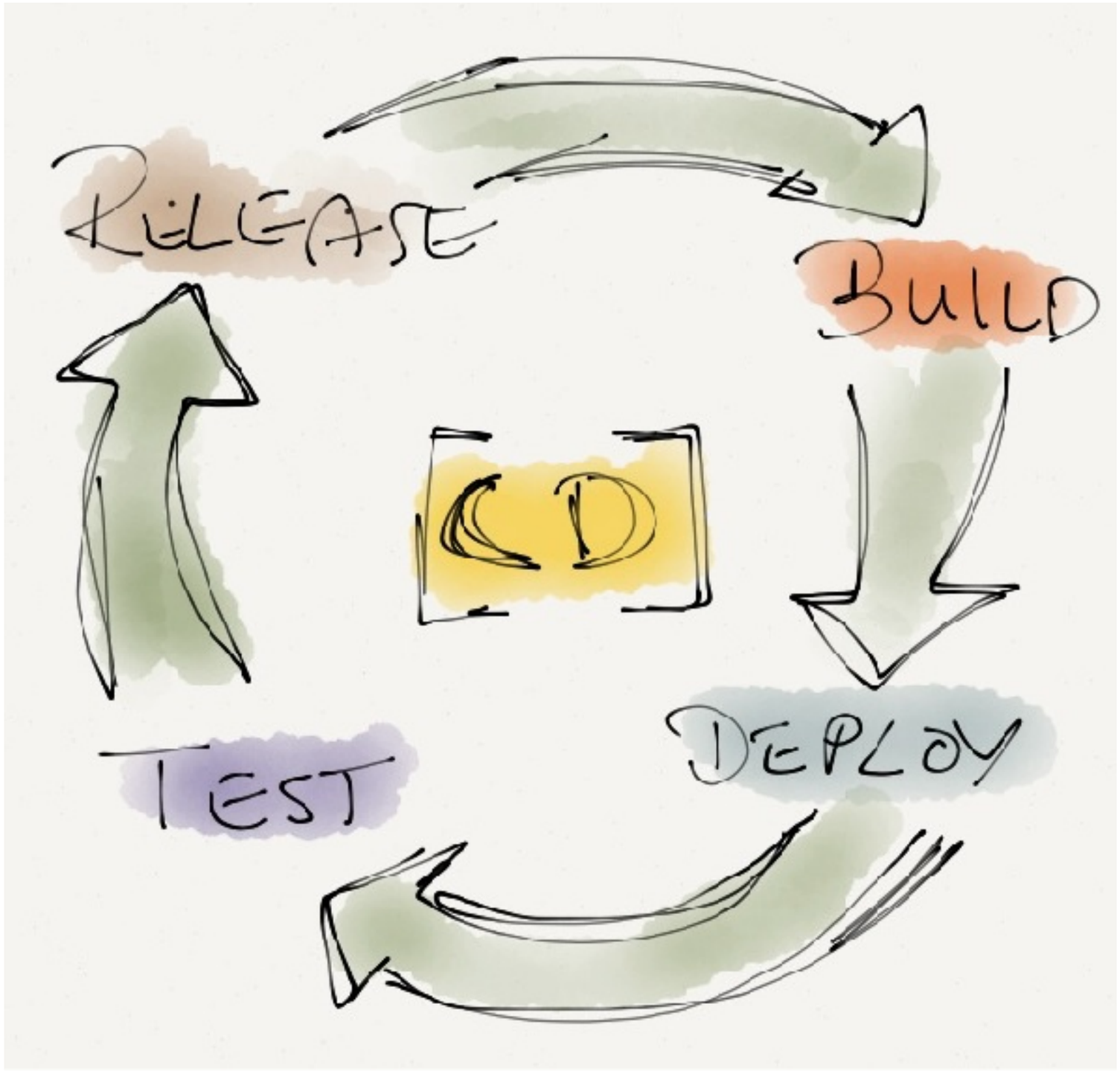




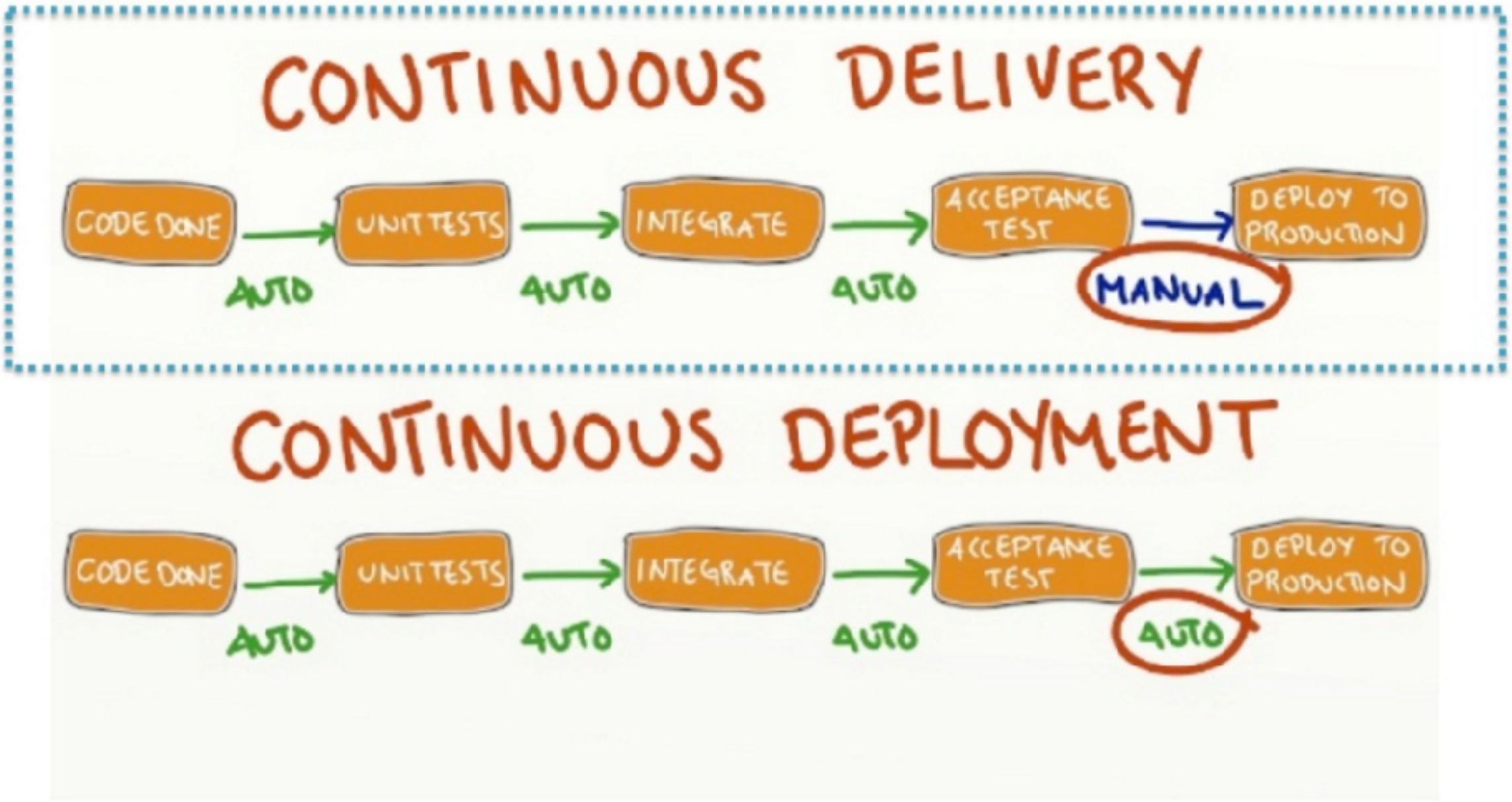
Continuous Integration



CD ?

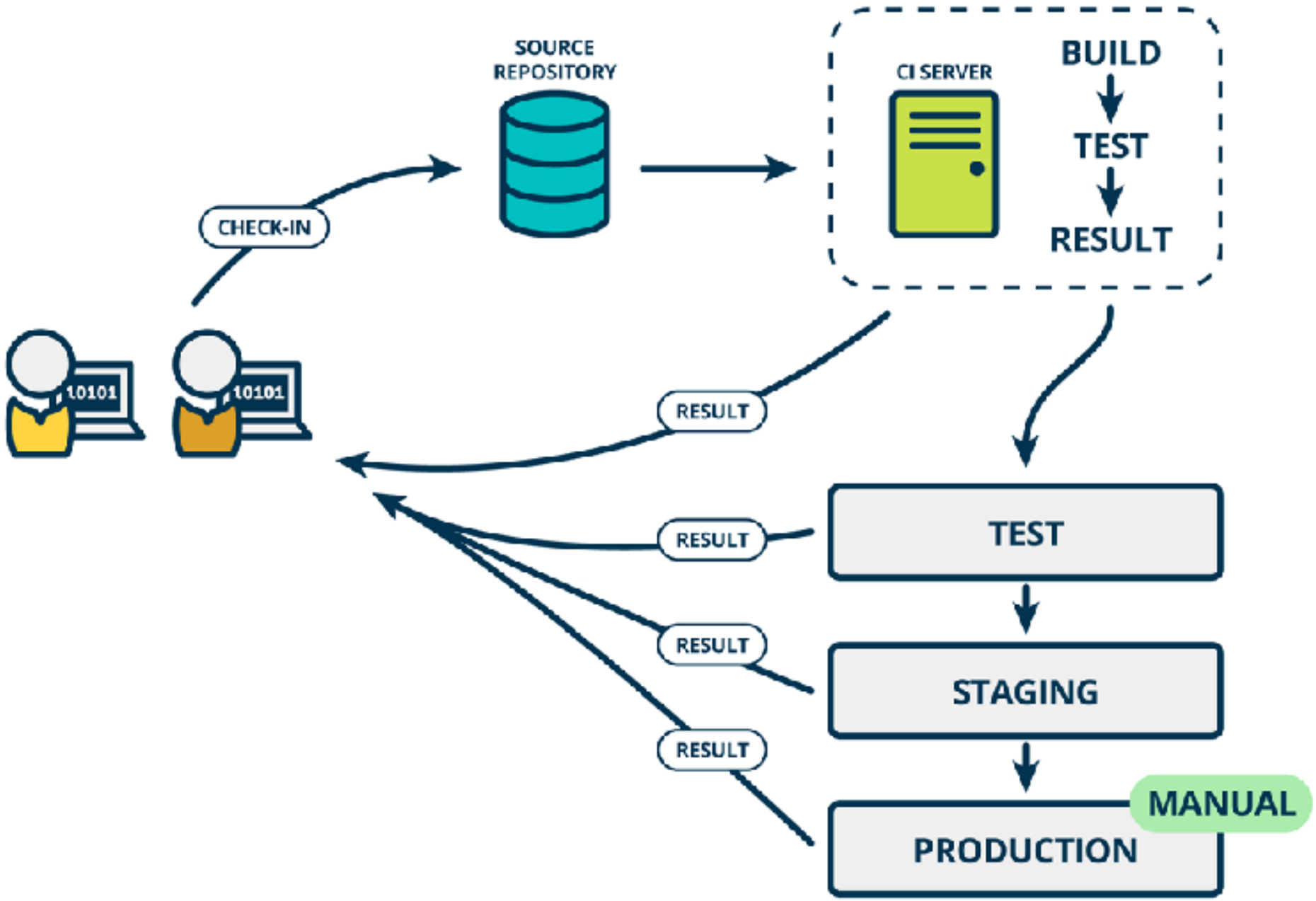


CD ?

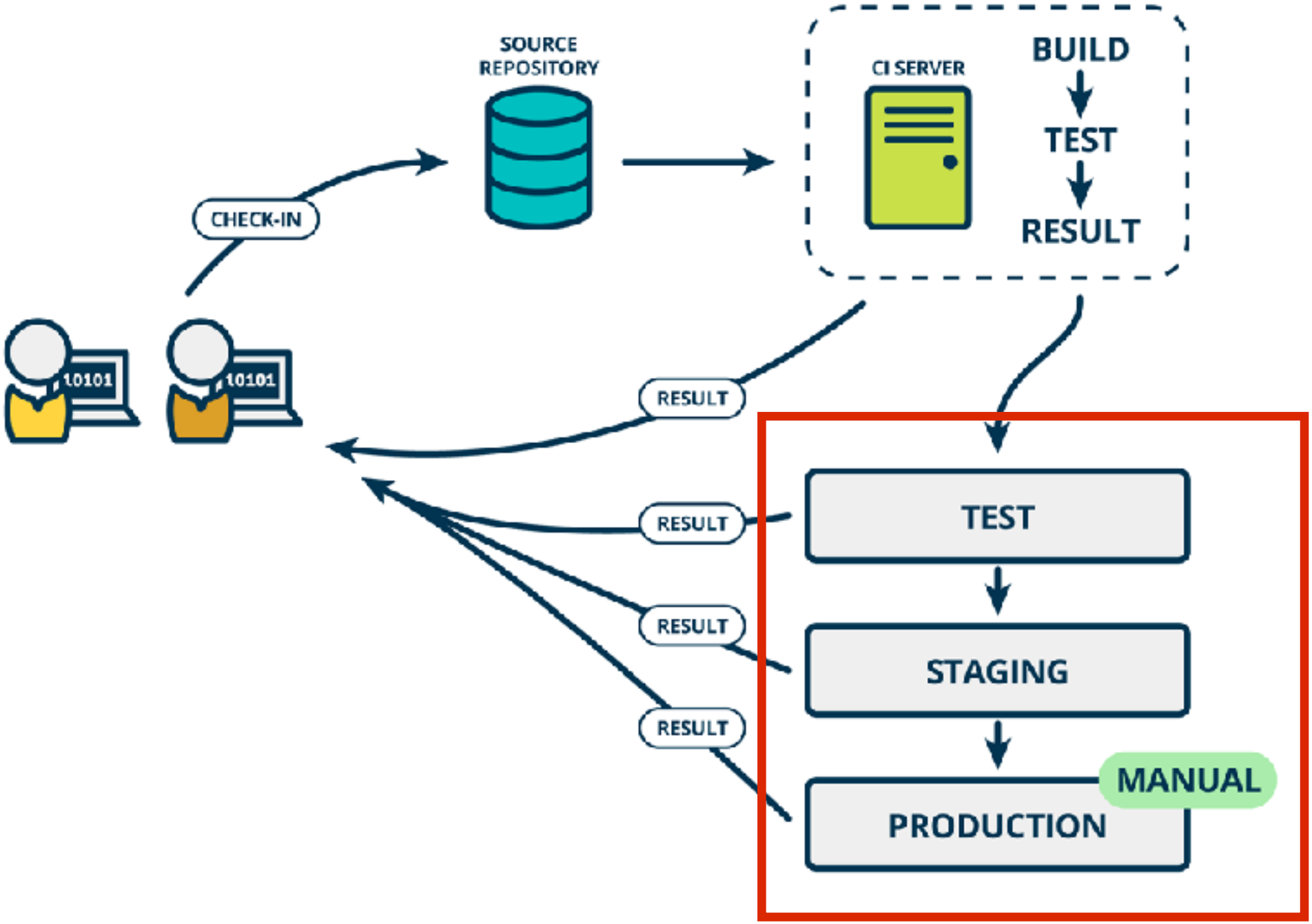


http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Continuous Delivery



Rise of DevOps



Continuous Integration

is a Software development **practices**

Practice 1

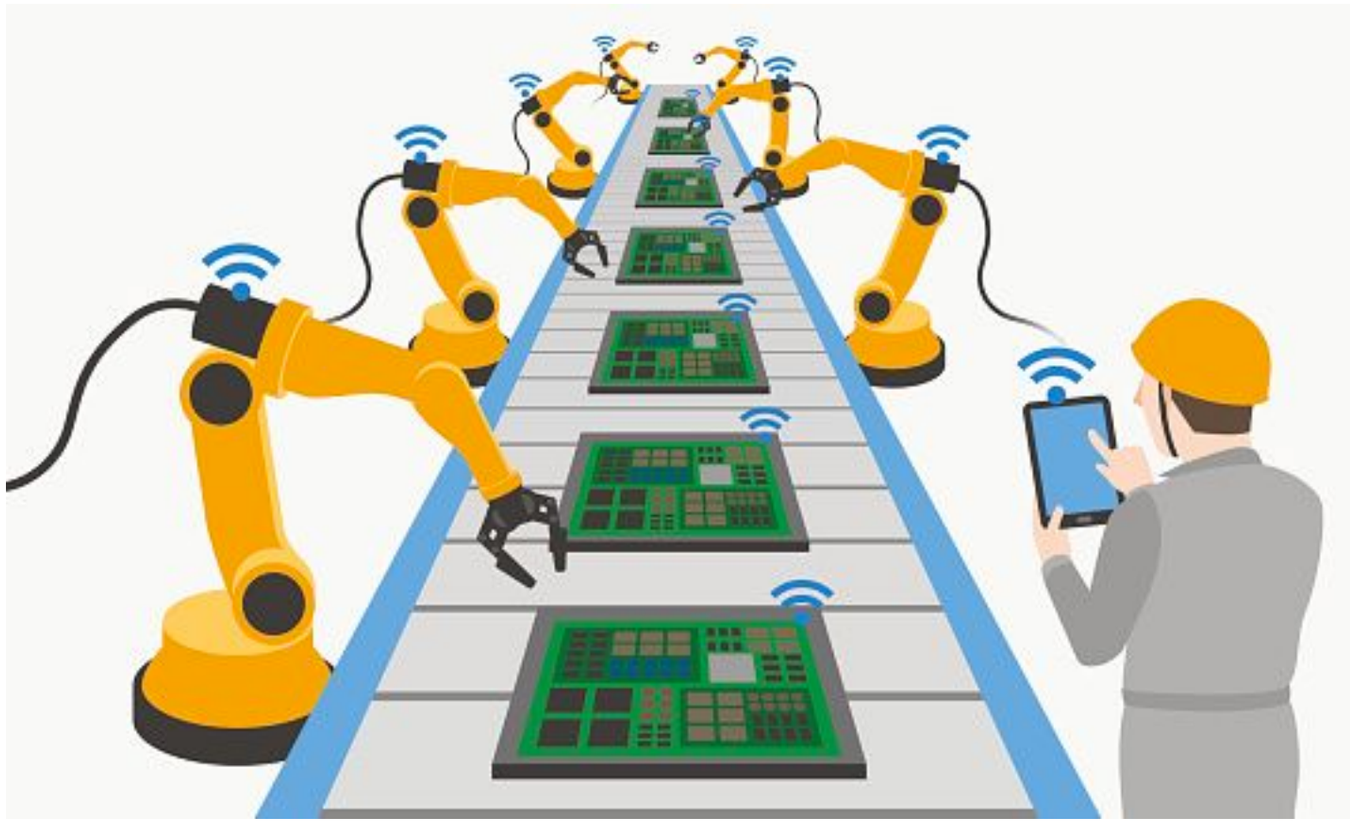
Maintain a single source repository

In general, you should store in source control everything you need to build anything

Practice 2

Automated the build

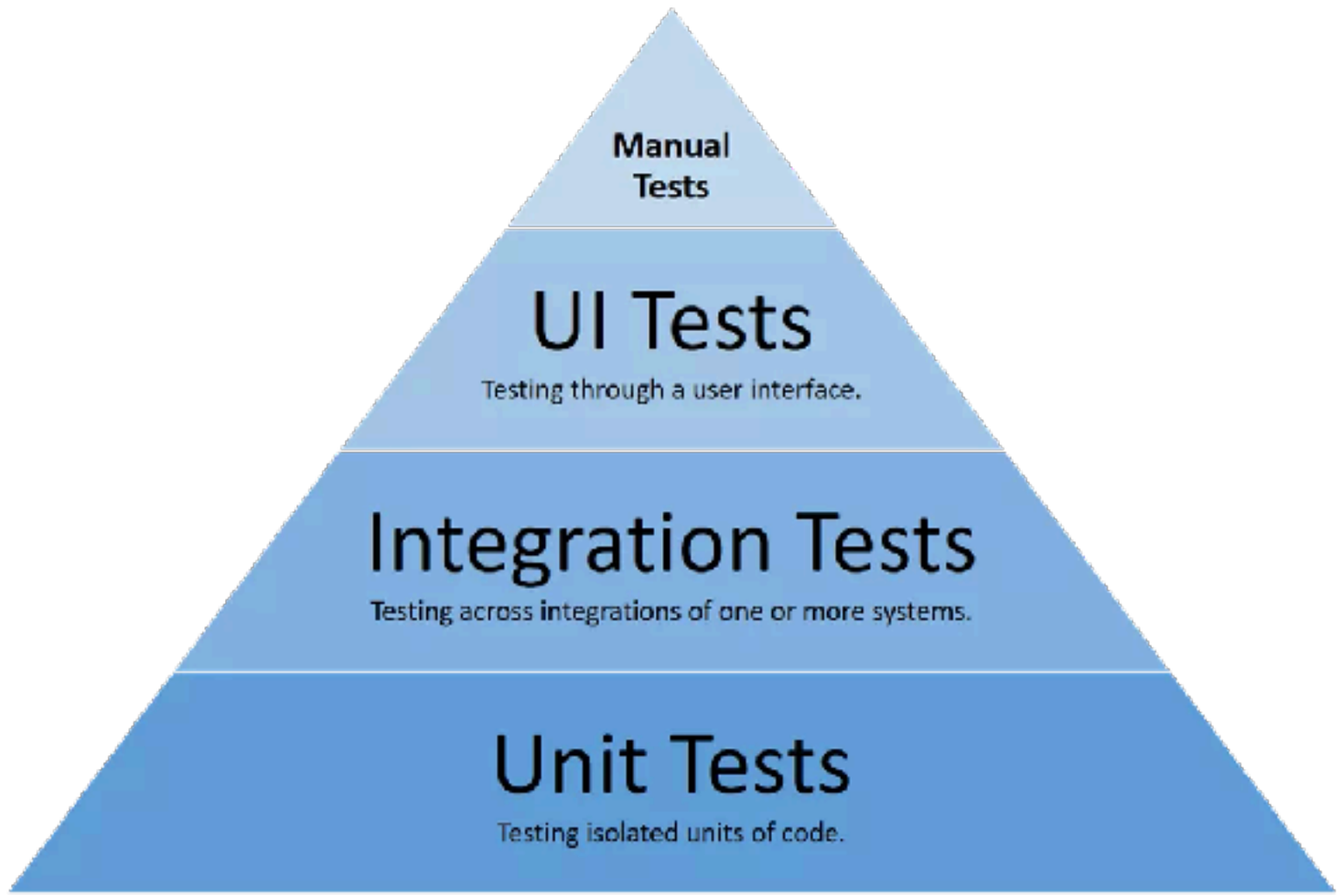
Automated environment for builds



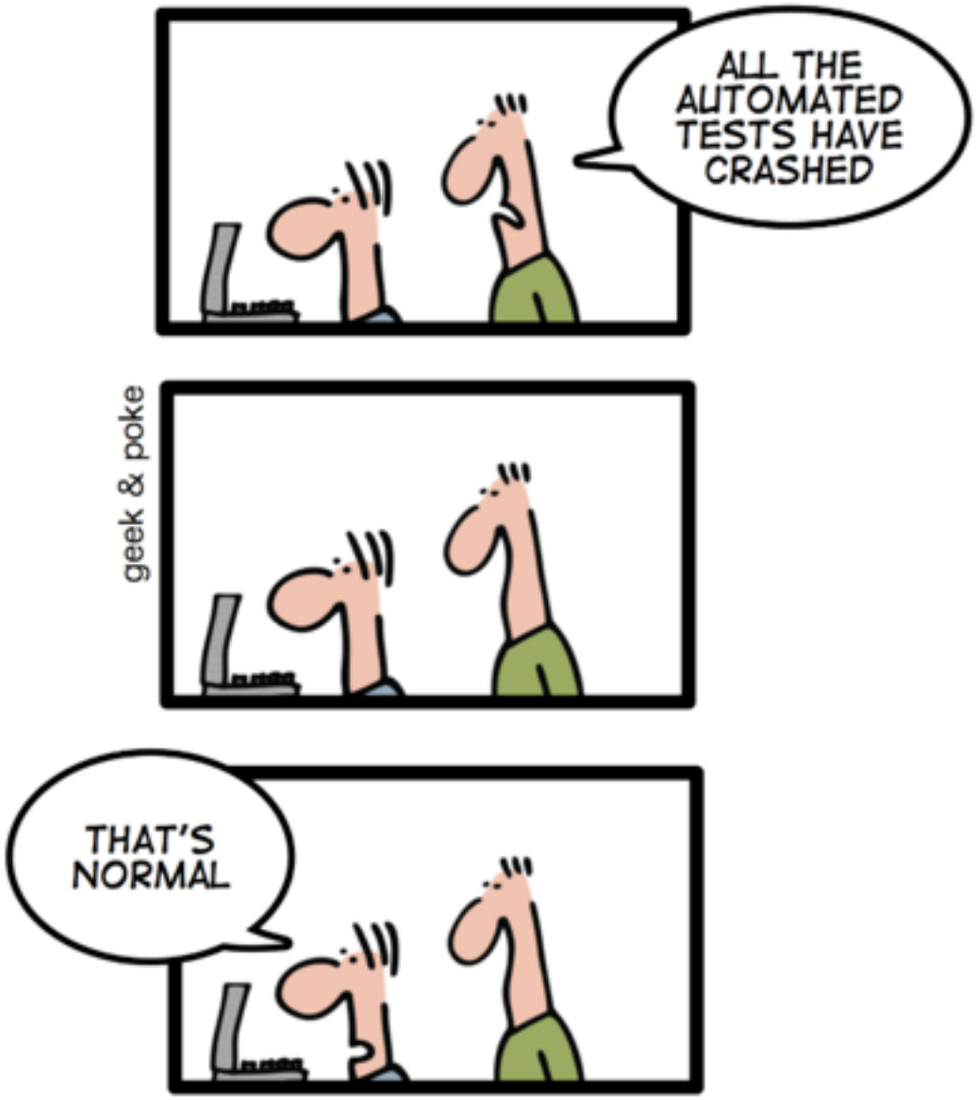
Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**



TODAY: CONTINUOUS INTEGRATION GIVES YOU THE COMFORTING FEELING TO KNOW THAT EVERYTHING IS NORMAL



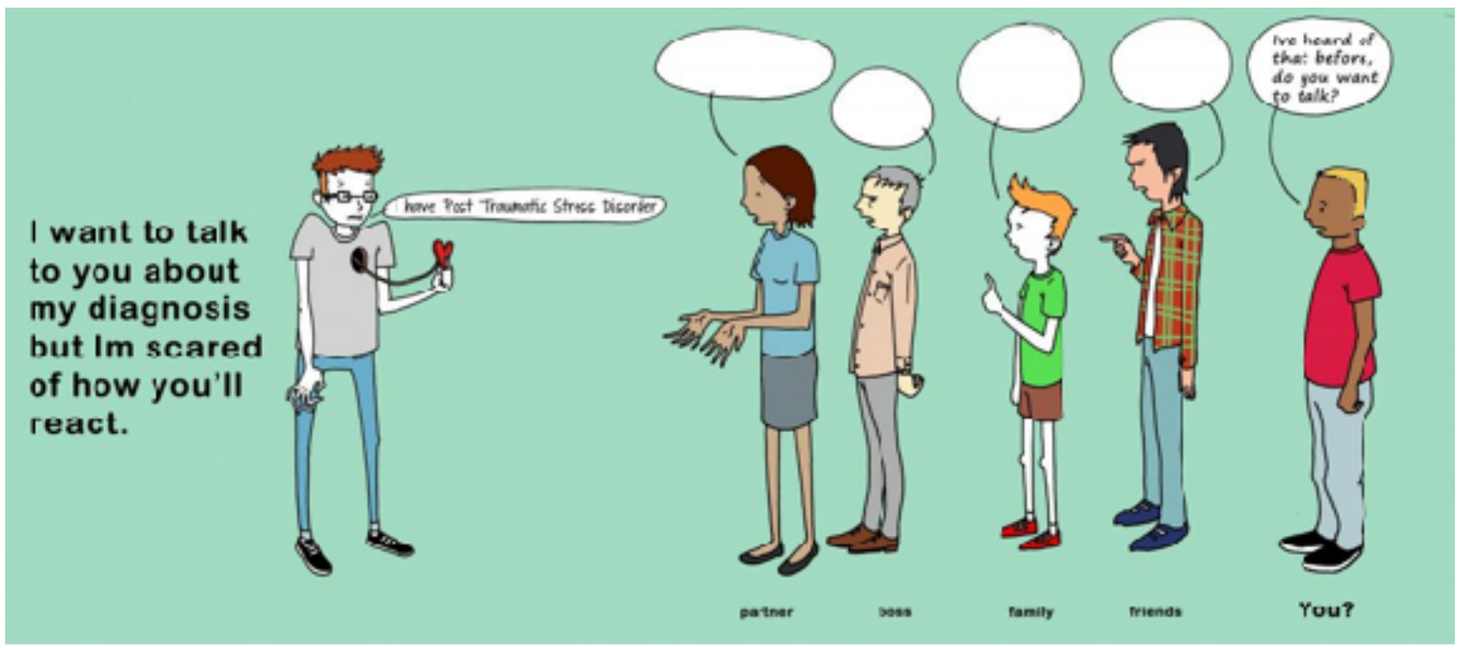
http://geekandpoke.typepad.com/

Practice 4

Everyone **commits** to the mainline everyday

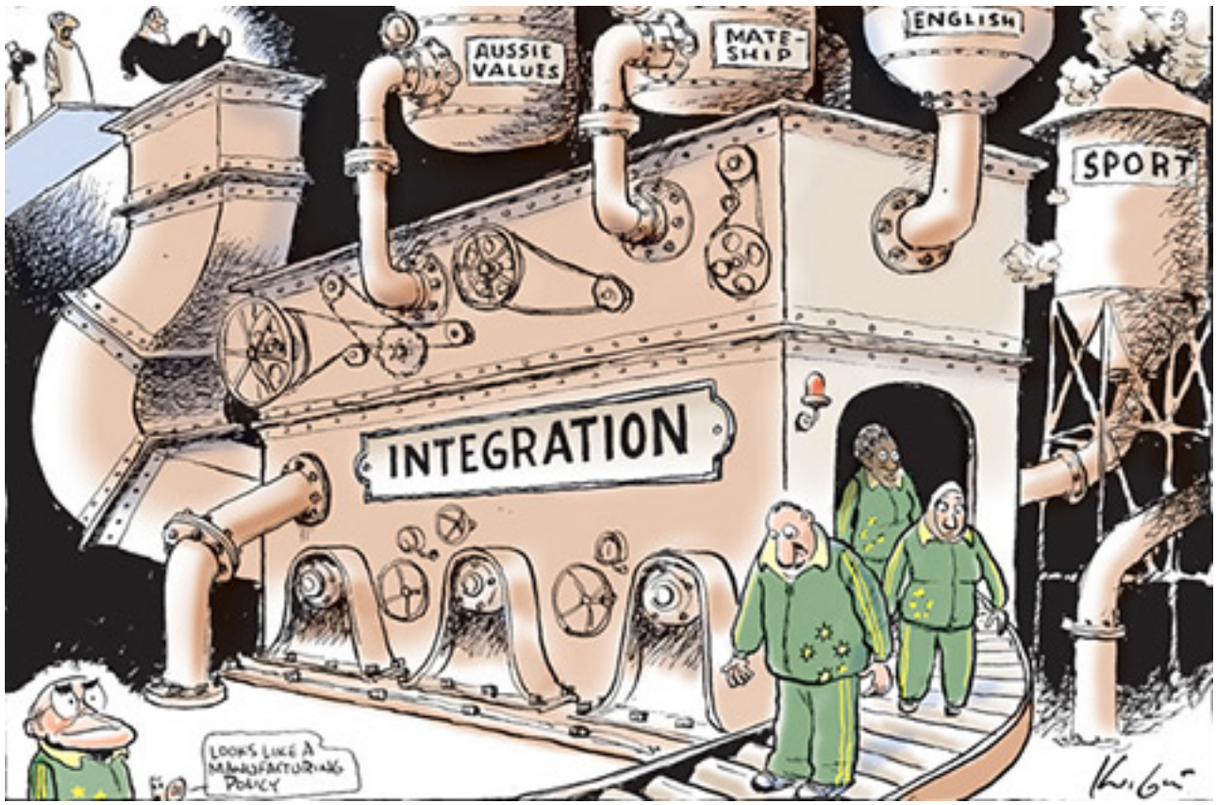
Integration is about **communication**

Integration allows developers to **tell** other developers



Practice 5

Every commits should build the mainline on an **Integration machine**



Nightly build is not enough for Continuous Integration

Practice 6

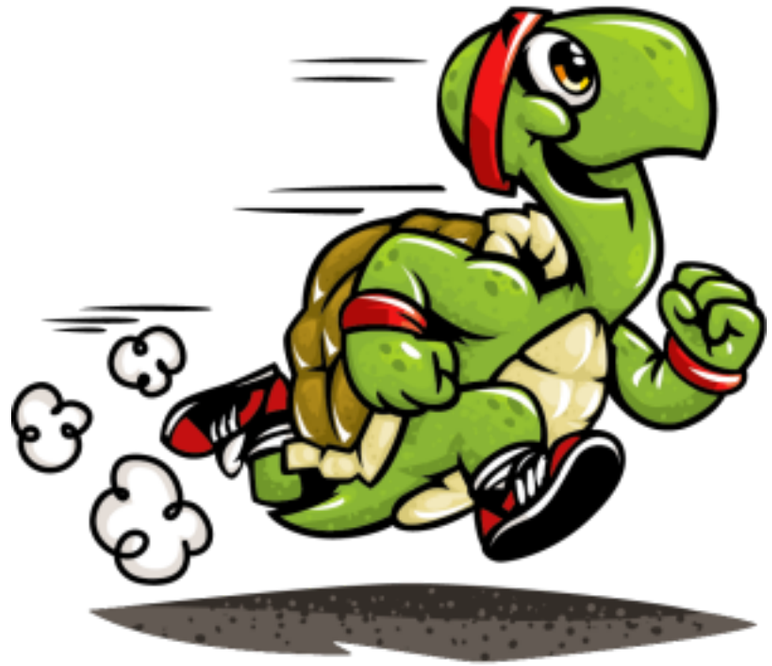
Fix broken builds **immediately**

“Nobody has a higher priority task than fixing the build”

Practice 7

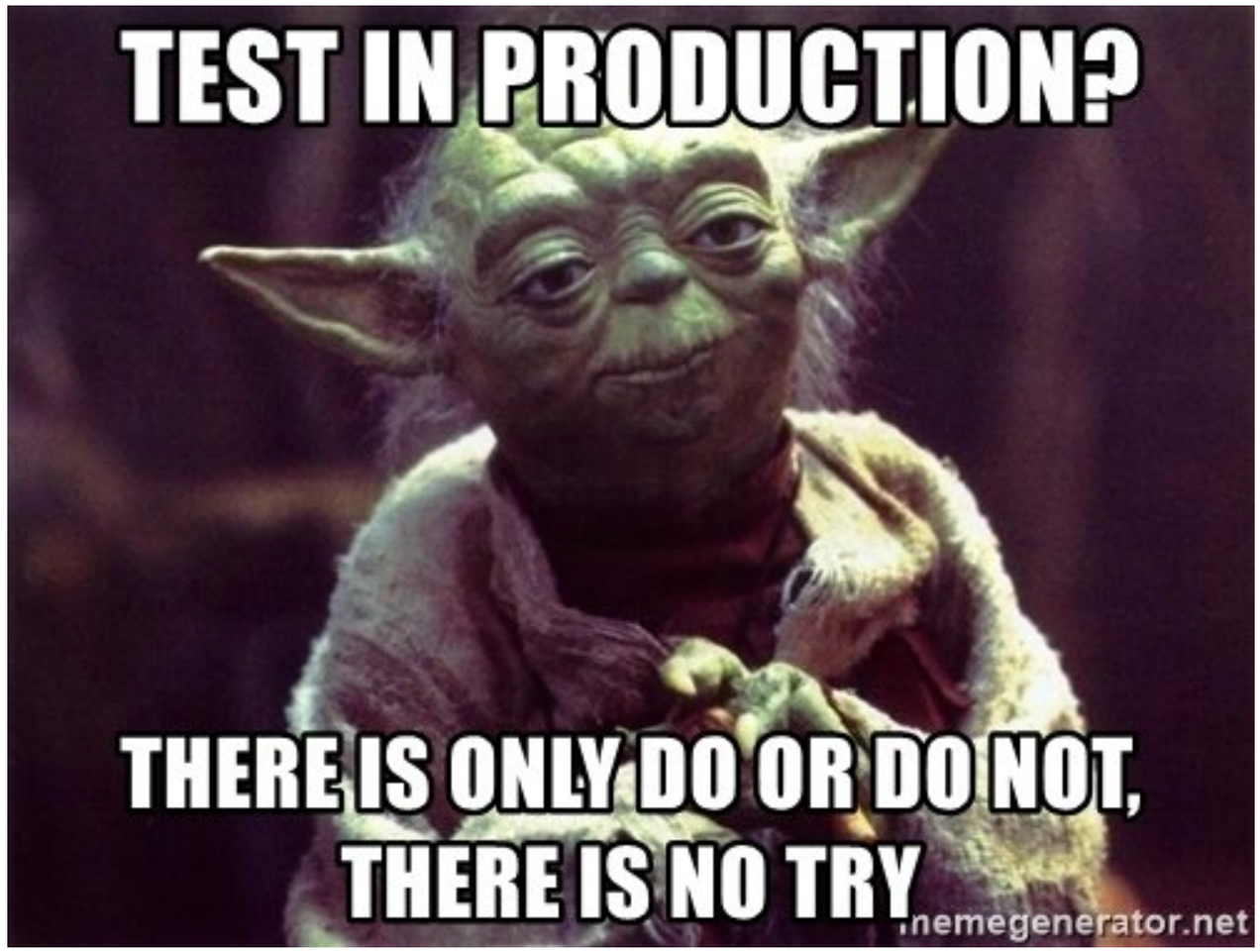
Keep the build **fast**

Continuous Integration is to provide rapid feedback



Practice 8

Test in clone of the **Production** environment



Practice 9

Make it easy for anyone to get
the latest executable

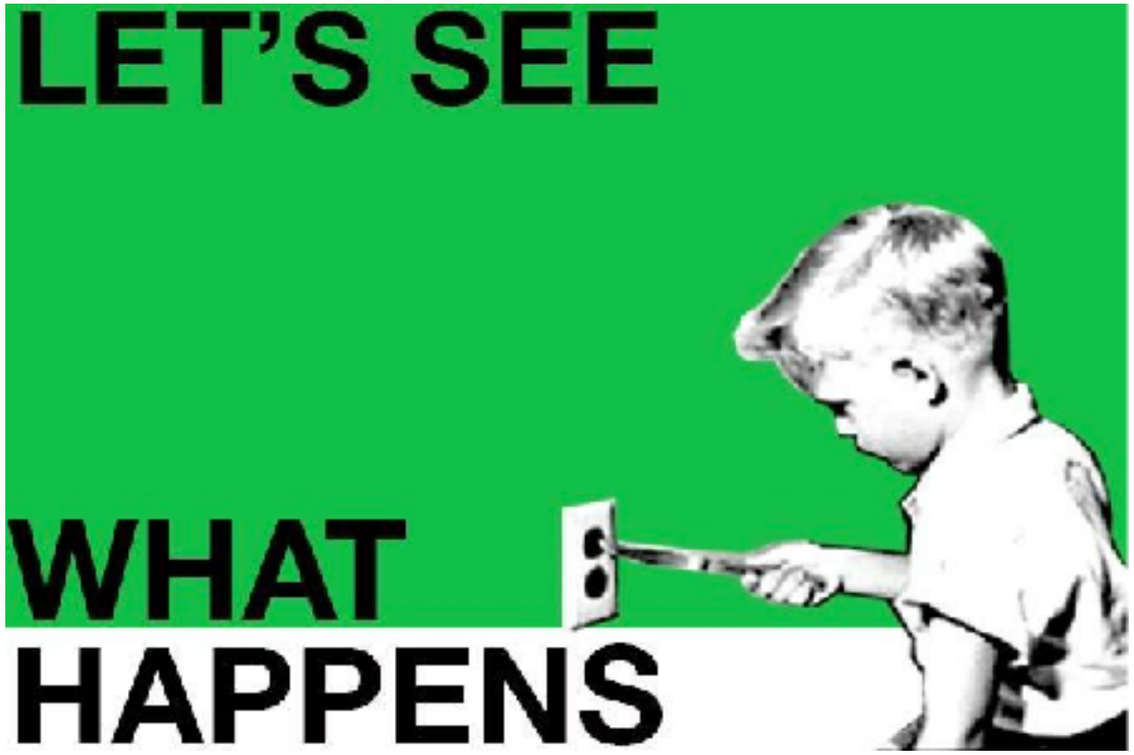
Make sure well known place where people can find

Practice 10

Everyone can see what's happening

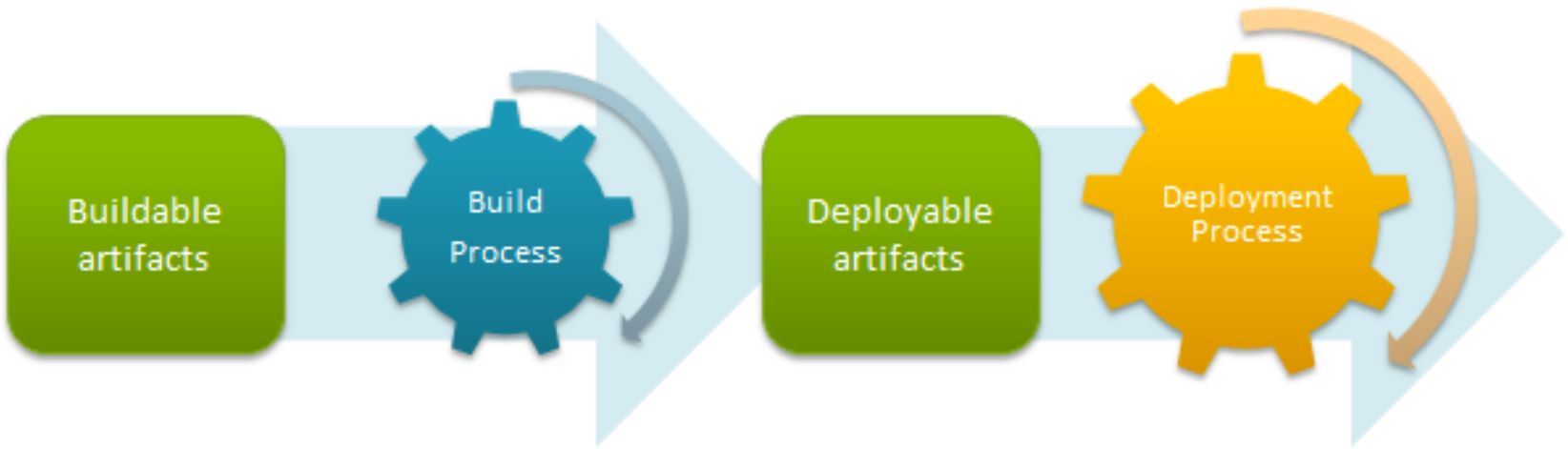
Easier to see the state of the system and changes

Show the good information



Practice 11

Automated deployment



How to achieve the CI ?

1. Use good version control

Local

Centralize

Distributed

2. Choose Branch strategy

Main only

Development isolation

Feature isolation

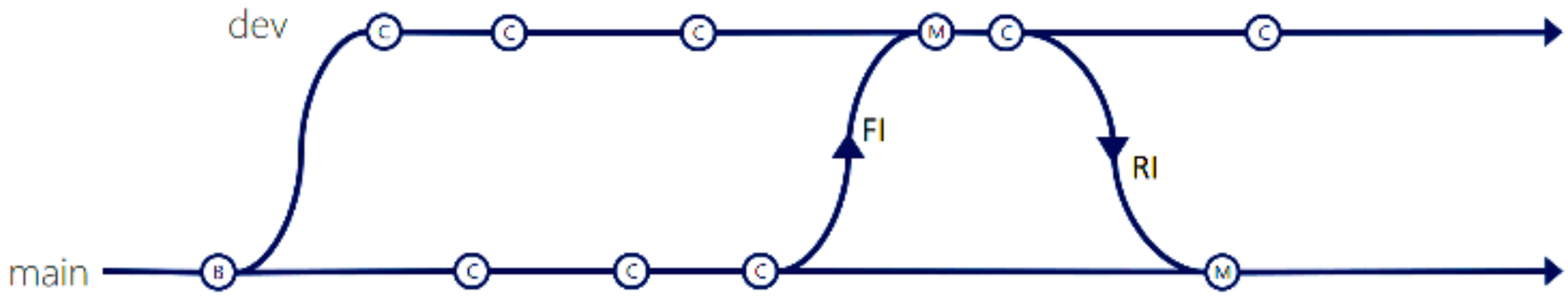
Release isolation

Service and Release isolation

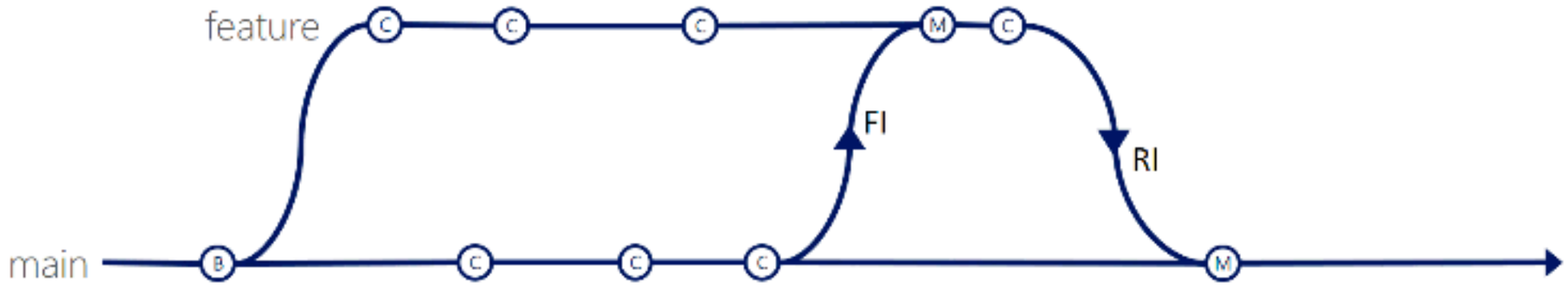
Service, Hotfix and Release isolation

Main only

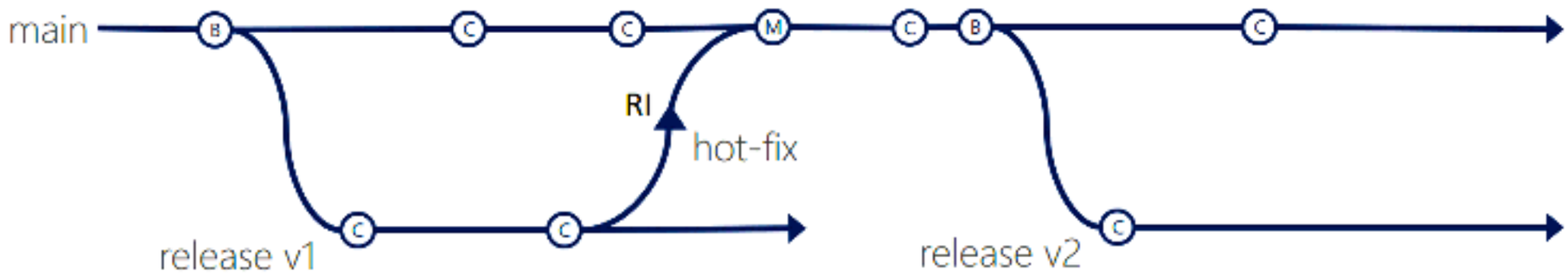
Development isolation



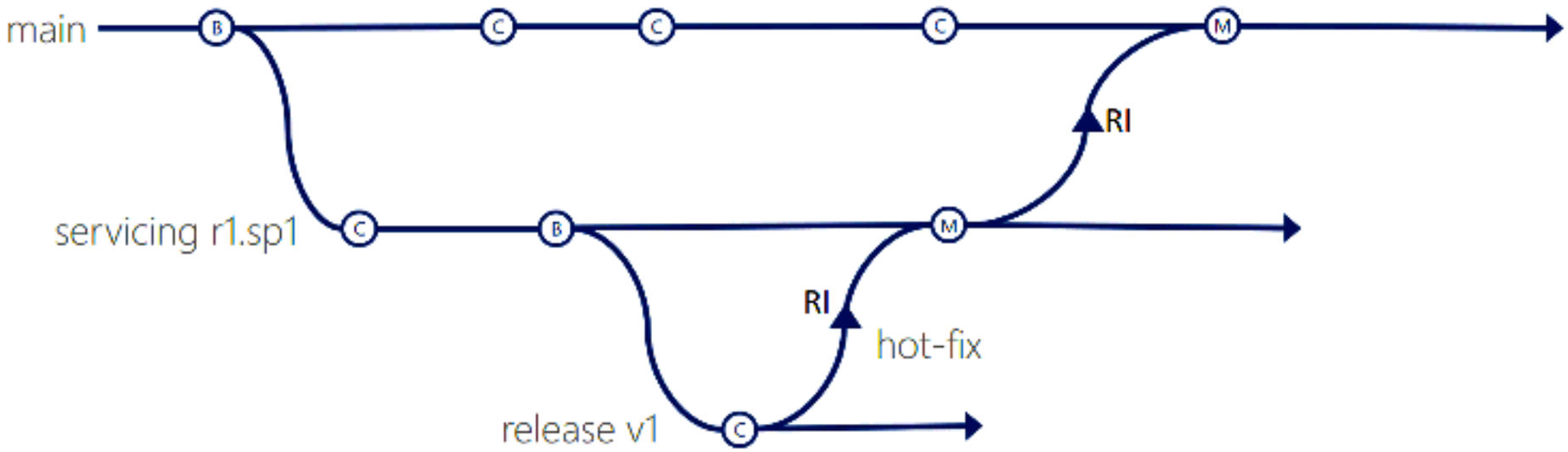
Feature isolation



Release isolation



Service and Release isolation



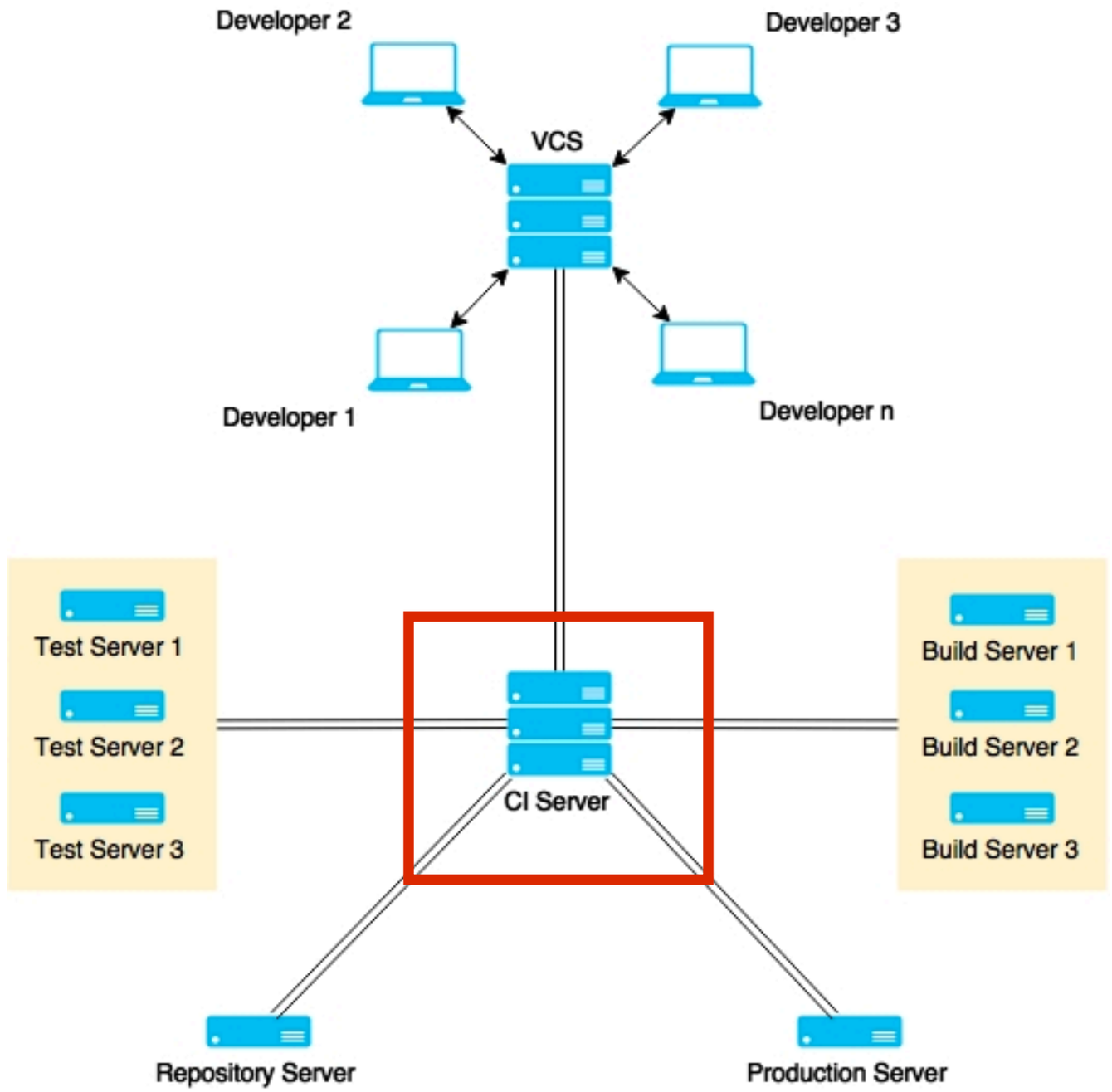
Service, Hotfix, Release isolation

Validate, Validate and Validate

Suggestion

Keeping branches short-lived, merge conflicts are keep to as few as possible

3. Use good CI tool



Jenkins

Bamboo

TeamCity

go

Visual Studio Team Foundation Server

Hudson

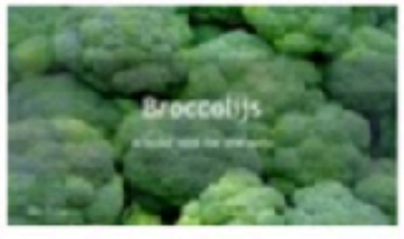
Travis

wercker

circleci

4. Use good build tool

- Javascript
  - Gulp, Grunt, Brocolli



- C#/.NET
  - Nant, MSBuild

- Java/JVM
  - Ant, Maven, Gradle, SBT, Leiningen

More …

Use static code analysis

Automated testing

Automated deployment

People discipline/habit

“Behind every successful agile project, there is a

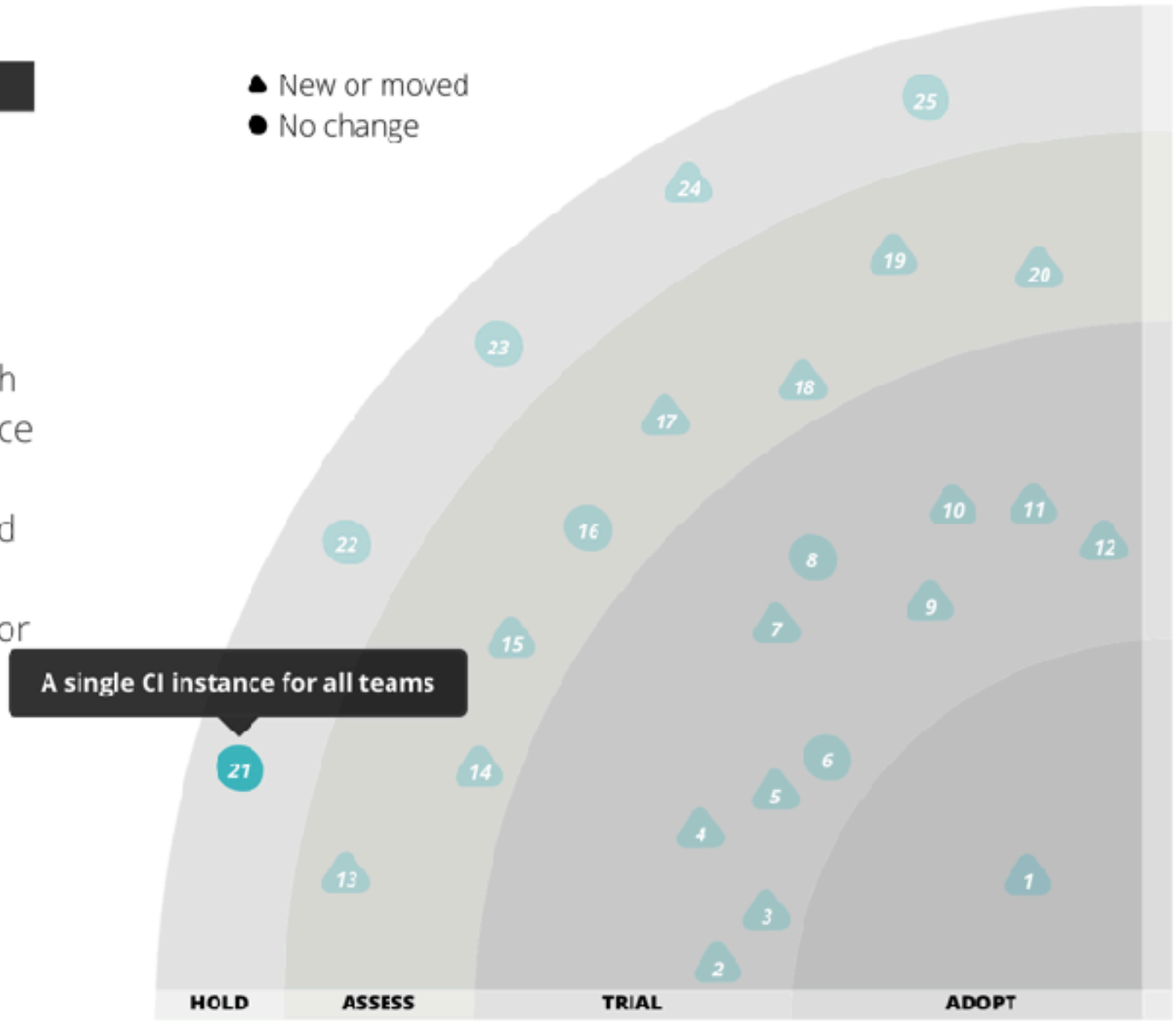
Continuous Integration Server”

Anti-pattern for CI Server

HOLD ?

21. A single CI instance for all teams

We're compelled to caution, again, against creating **a single CI instance for all teams**. While it's a nice idea in theory to consolidate and centralize Continuous Integration (CI) infrastructure, in reality we do not see enough maturity in the tools and products in this space to achieve the desired outcome. Software delivery teams which must use the centralized CI offering regularly have long delays depending on a central team to perform minor configuration tasks, or to troubleshoot problems in the shared infrastructure and tooling. At this stage, we continue to recommend that organizations limit their centralized investment to establishing patterns, guidelines and support for delivery teams to operate their own CI infrastructure.



https://www.thoughtworks.com/radar/techniques

Let’s start with CI/CD

Application and framework to manage and monitor the executable of **repeated tasks**

Jenkins

https://jenkins.io/

Centralize Continuous Integration Server

Jenkins

https://jenkins.io/

Why Jenkins ?

Easy !!

Extensible

Scalable

Opensource

Large community

Lot of plugins

Who use Jenkins ?

We thank the following organizations for their major commitments to support the Jenkins project.

Microsoft · cloudbees · OSL OPEN SOURCE LAB

rackspace redhat

We thank the following organizations for their support of the Jenkins project through free and/or open source licensing programs.

Atlassian Datadog JFrog Mac Cloud PagerDuty XMission

https://jenkins.io/

Hardware requirements

For Jenkins server

RAM +2GB

More CPU

More Disk

Installation

Jenkins in containers

Apache Tomcat

Jetty

JBoss

Websphere

WebLogic

Glassfish

Download Jenkins

https://jenkins.io/

Use Long Term Support (LTS)

Getting started with Jenkins

The Jenkins project produces two release lines, LTS and weekly. Depending on your organization's needs, one may be preferred over the other.

Both release lines are distributed as `.war` files, native packages, installers, and Docker containers.

Long-term Support (LTS)

LTS (Long-Term Support) releases are chosen every 12 weeks from the stream of regular releases as the stable release for that time period. Learn more...

Changelog | Upgrade Guide | Past Releases

Deploy Jenkins 2.46.3

Deploy to Azure

Download Jenkins 2.46.3 for:

- Docker
- FreeBSD

Weekly

A new release is produced weekly to deliver bug fixes and features to users and plugin developers.

Changelog | Past Releases

Download Jenkins 2.65 for:

- Arch Linux
- Docker
- FreeBSD
- Gentoo

Start Jenkins

$java -jar jenkins.war

Default port of server is **8080**

```
 org.eclipse.jetty.server.AbstractConnector doStart
ector@3e2fc448{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}
 org.eclipse.jetty.server.Server doStart
```

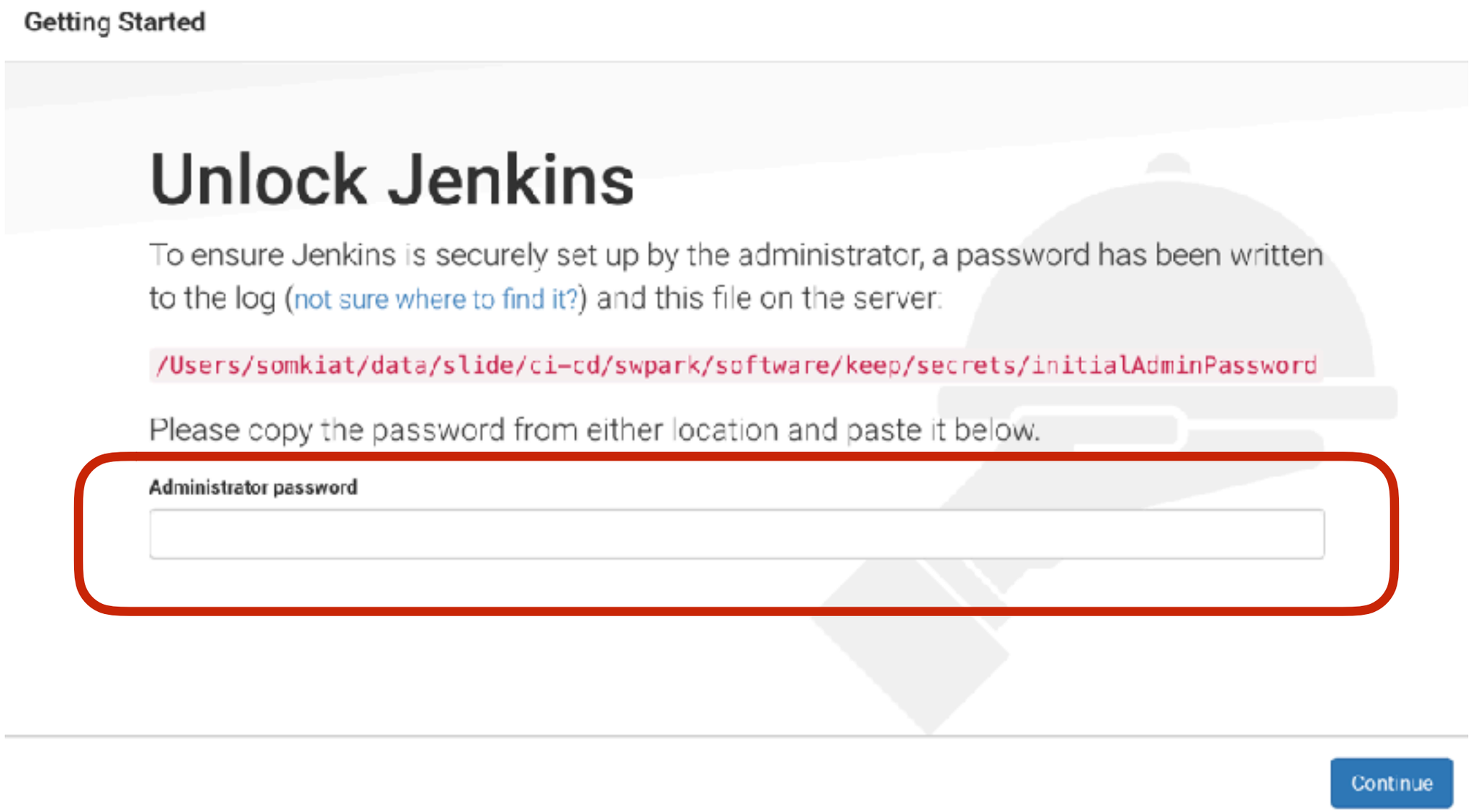
```
 winstone.Logger logInternal
gine v4.0 running: controlPort=disabled
 jenkins.InitReactorRunner$1 onAttained
ion
 jenkins.InitReactorRunner$1 onAttained

 jenkins.InitReactorRunner$1 onAttained
```

Change port of Jenkins

$java -jar jenkins.war **--httpPort=<port>**

Open in browser

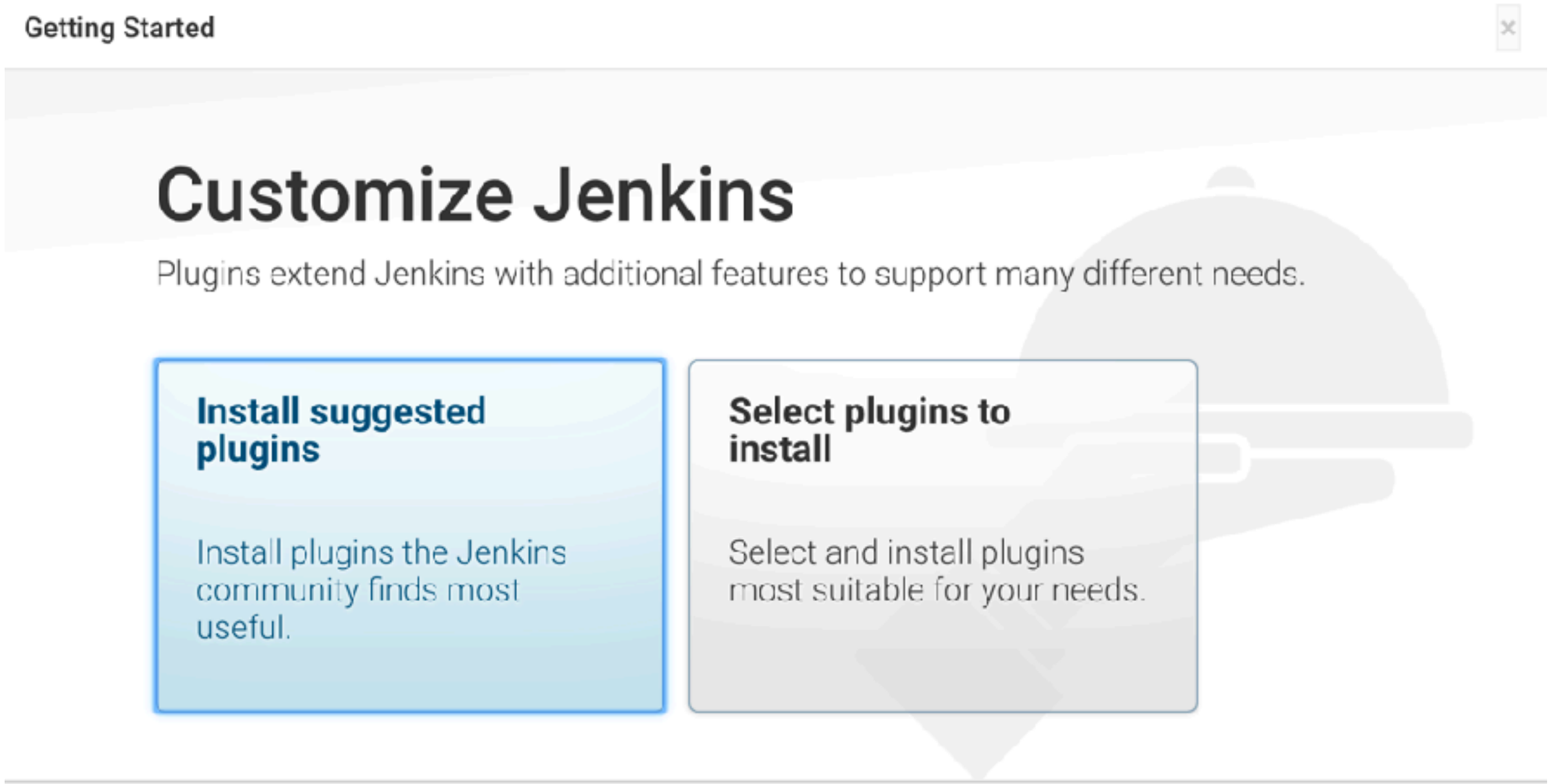
http://localhost:8080



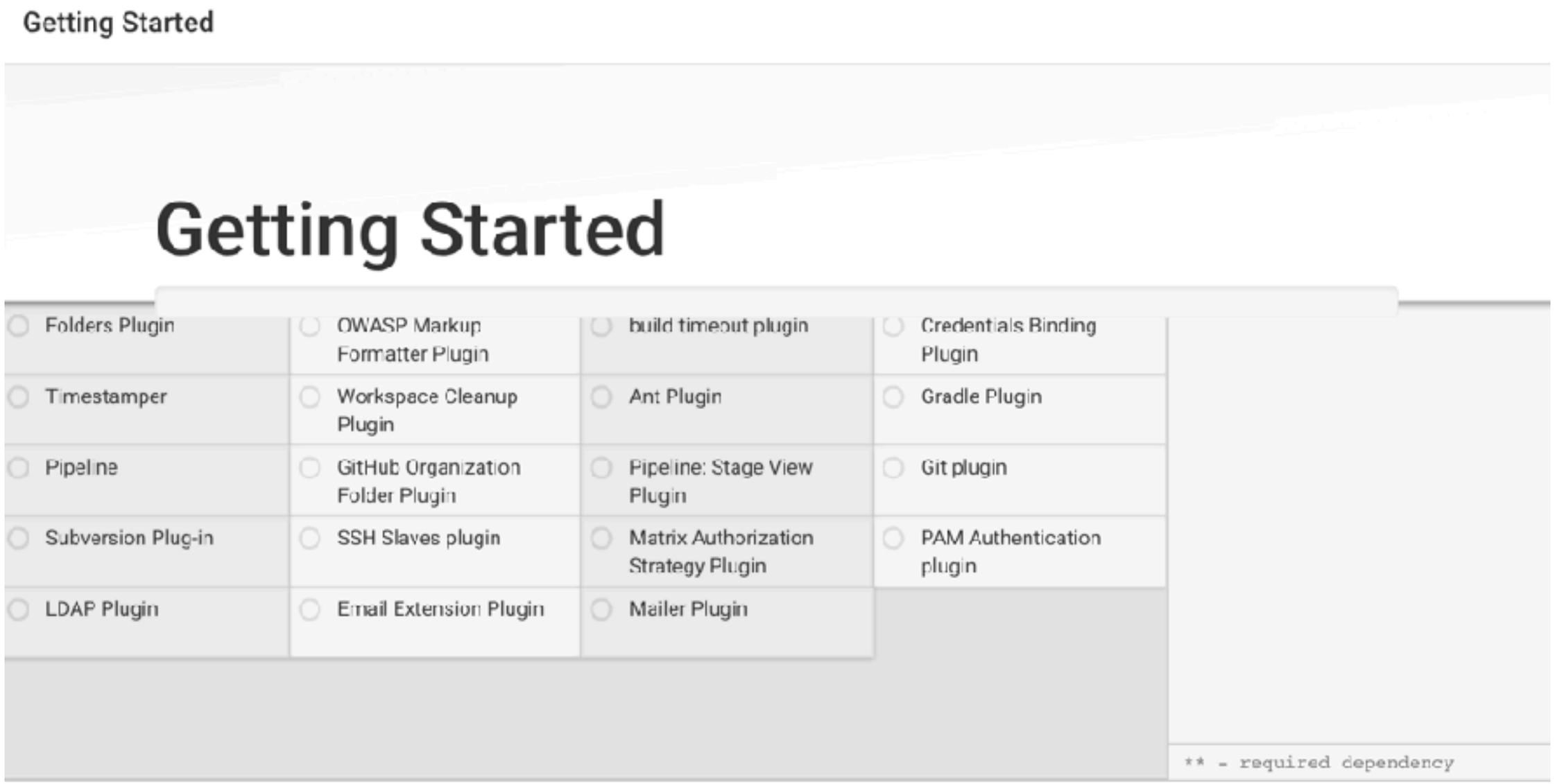
Copy password from console

```
*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been creat
.
Please use the following password to proceed to installation:

a4b3a5231b8048419192d0c5afd3fce8

This may also be found at: /Users/somkiat/data/slide/ci-cd/swpa
itialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

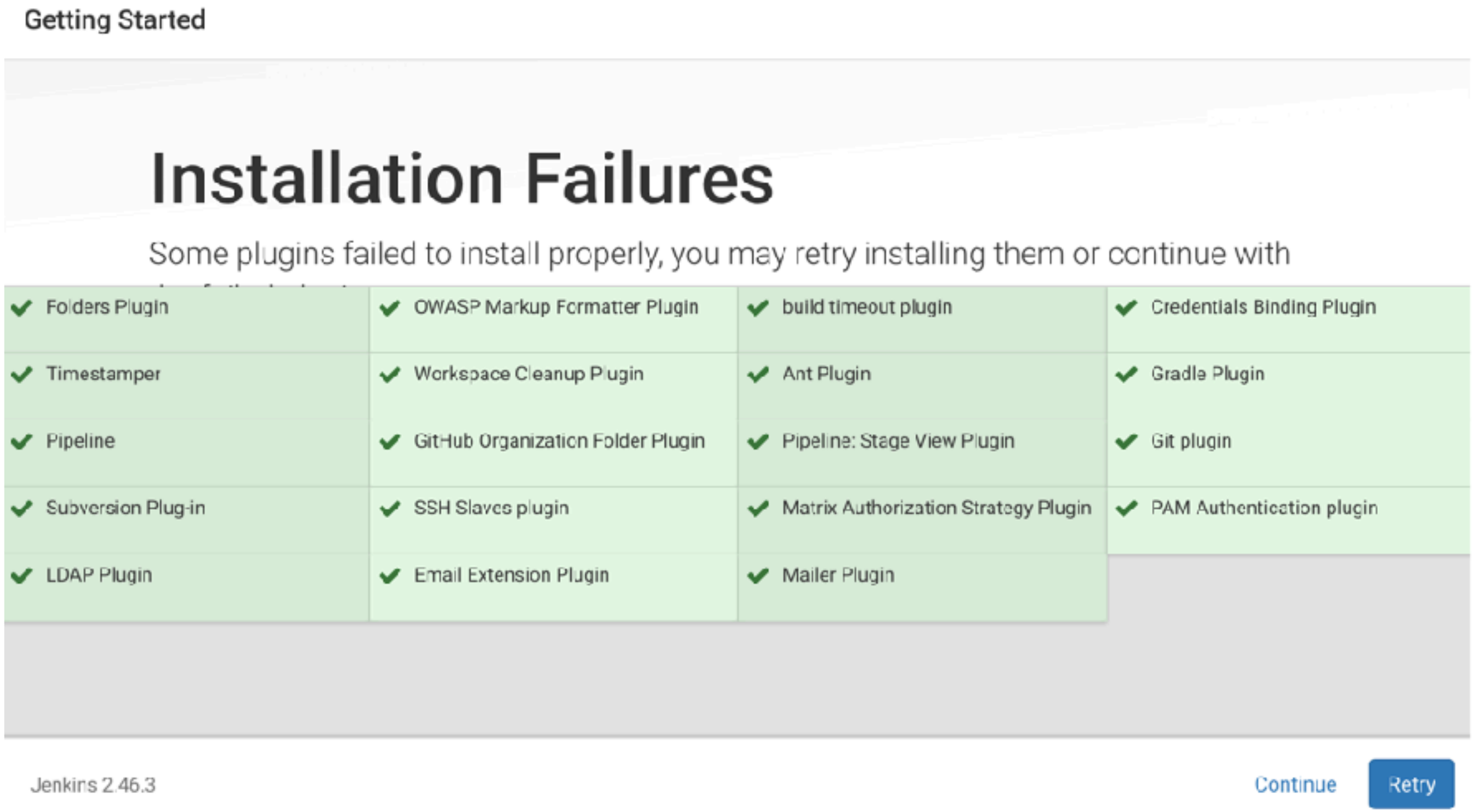
Customize plugins



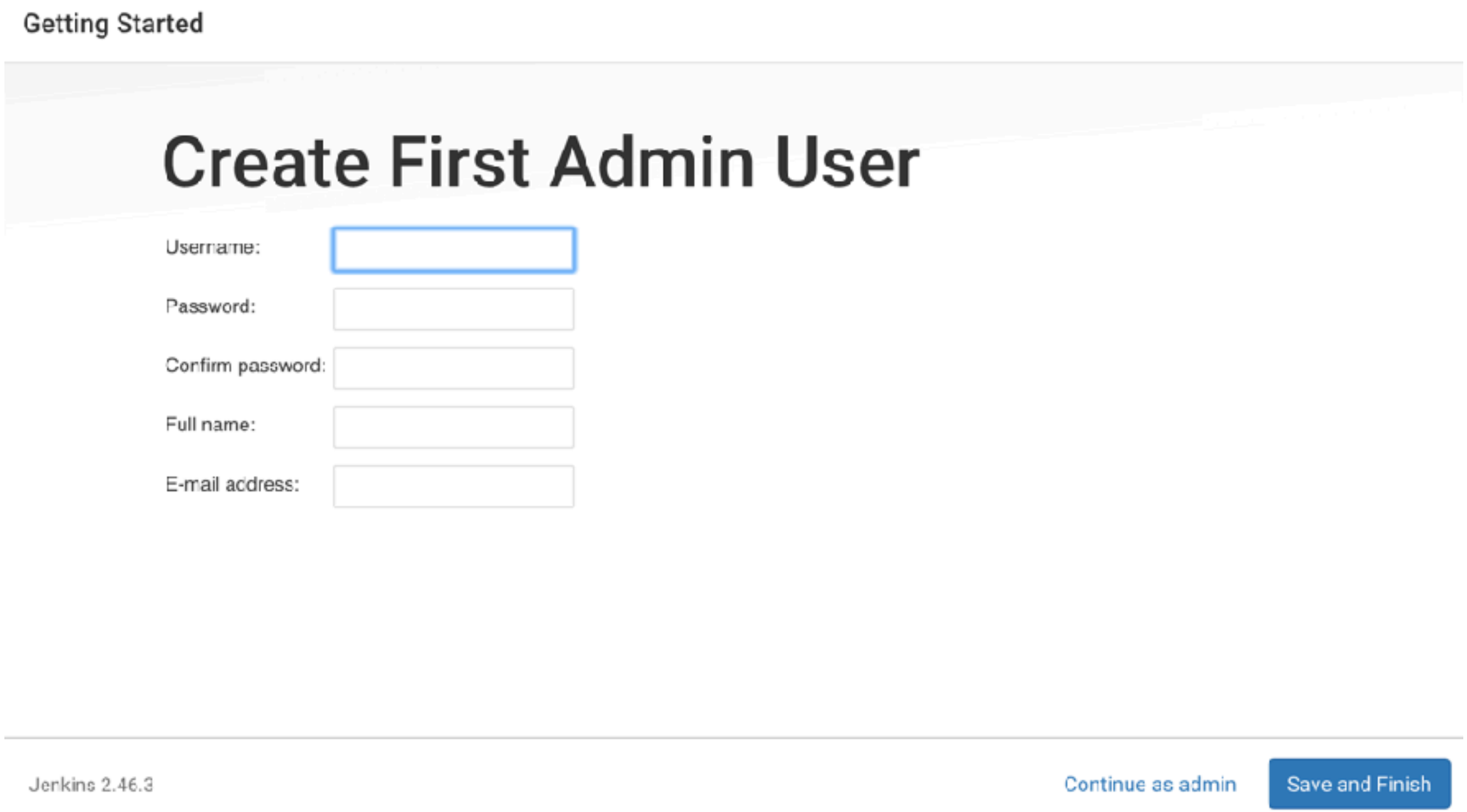
Waiting



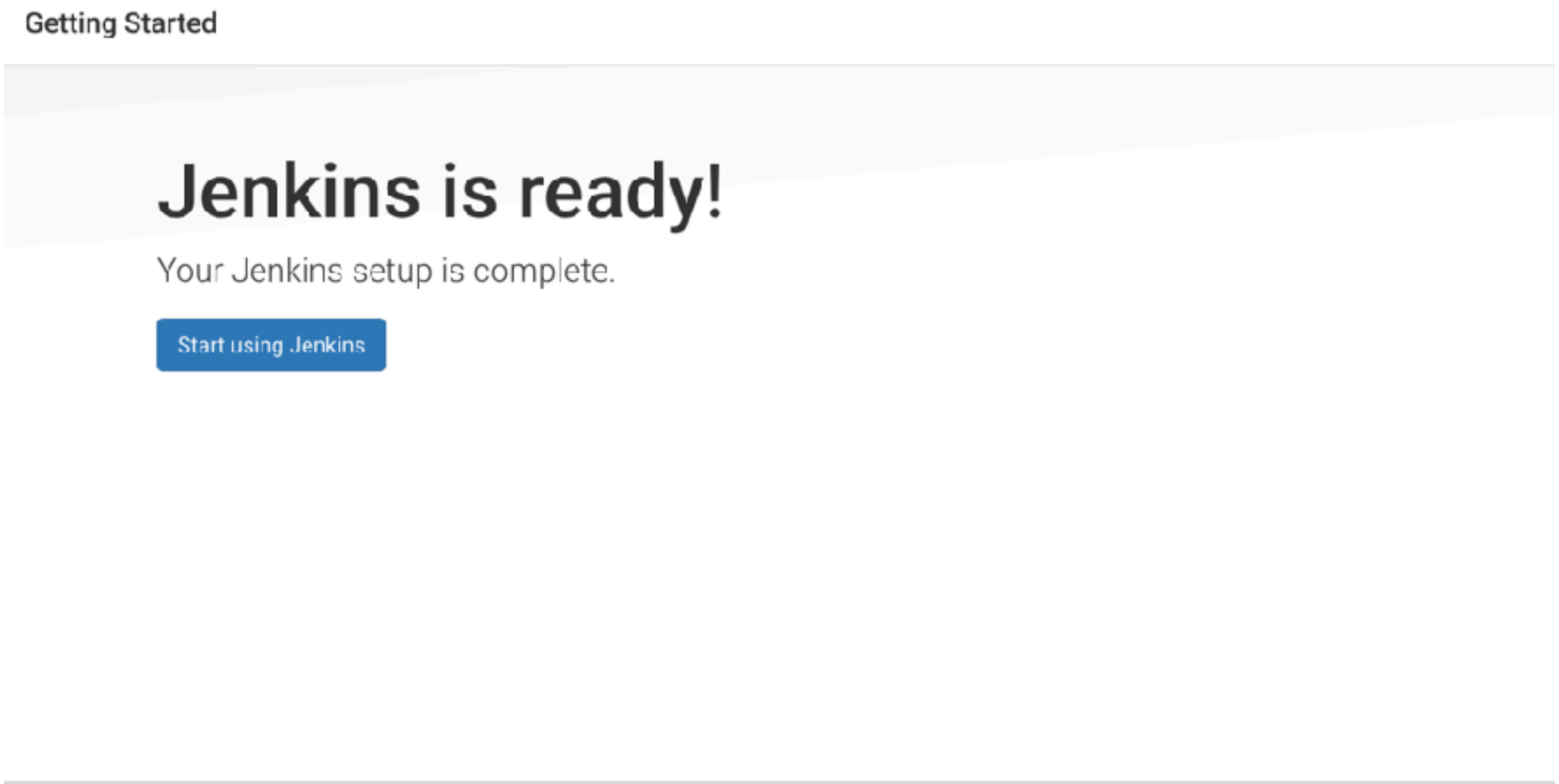
Success



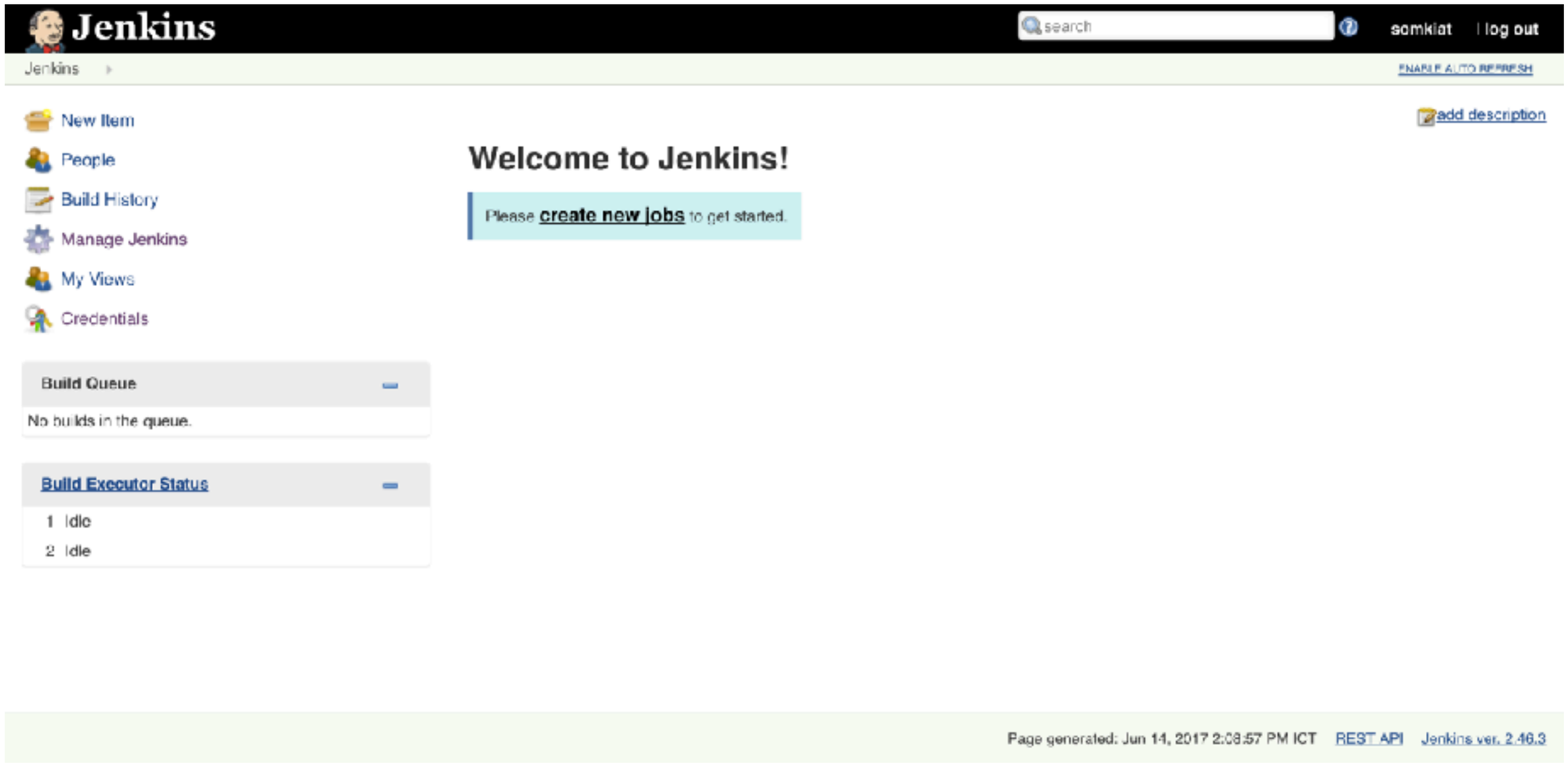
Create a new user



Ready to use



Welcome to Jenkins



About Jenkins's HOME

Default of **JENKINS_HOME** is

<path of user>/.jenkins

About Jenkins's HOME

Data in JENKINS_HOME

```
/Users/somkiat/.jenkins
├── config.xml
├── failed-boot-attempts.txt
├── hudson.model.UpdateCenter.xml
├── jenkins.CLI.xml
├── jenkins.install.UpgradeWizard.state
├── jobs
├── logs
├── nodeMonitors.xml
├── nodes
├── plugins
├── queue.xml
├── queue.xml.bak
├── secret.key
├── secret.key.not-so-secret
├── secrets
├── updates
├── userContent
├── users
└── war
```

About Jenkins's HOME

File and Folder name	Description
config.xml	All about configuration
jobs	Keep all jobs/project
plugins	Keep all plugins
nodes	Keep all nodes
logs	Keep all logs

Change Jenkins's HOME

For Windows

set JENKINS_HOME=<your path>

For Linux/Mac

export JENKINS_HOME=<your path>

try to restart Jenkins …

Disable Jenkins's security

Set useSecurity=false

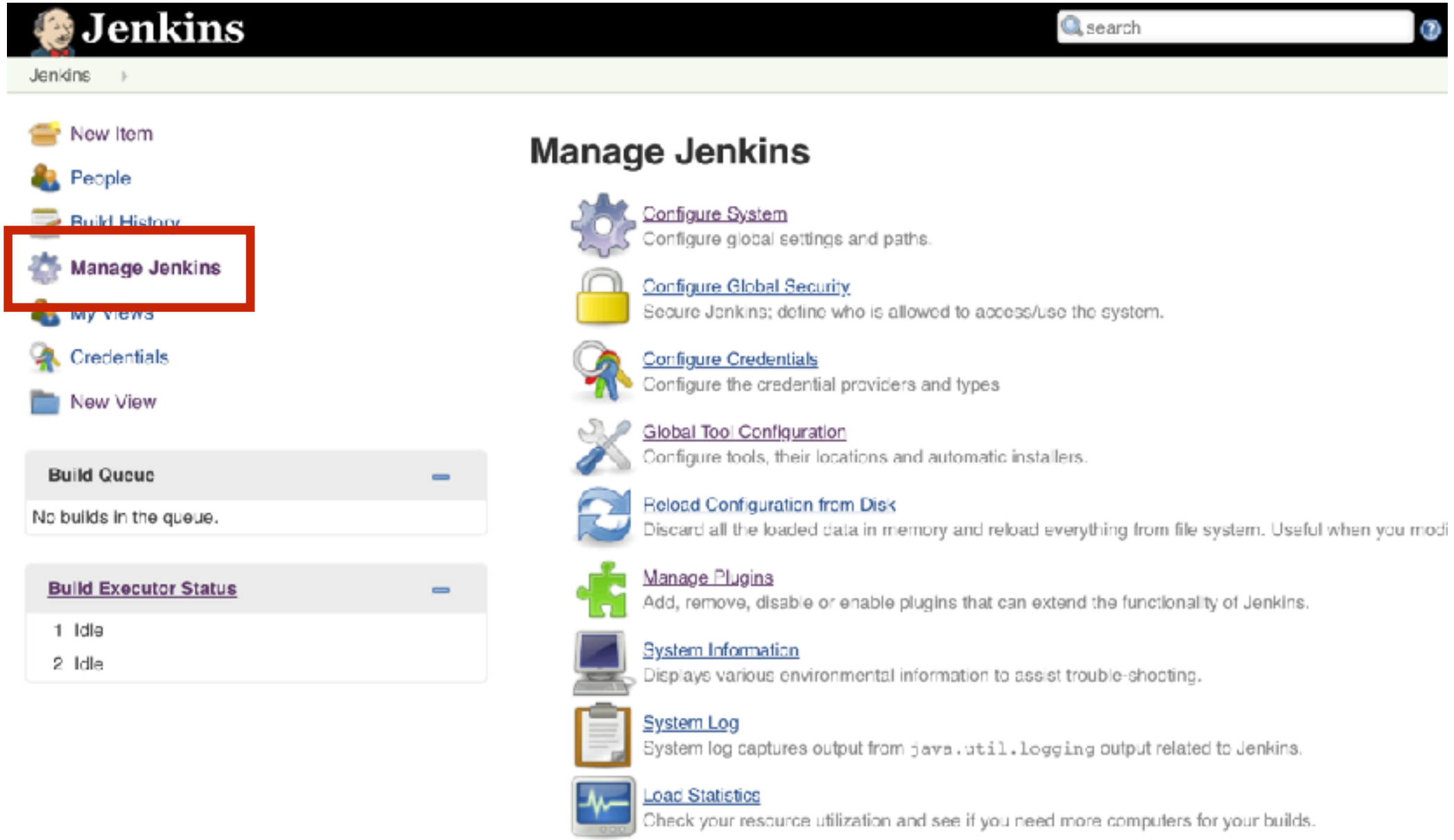
<JENKINS HOME>/config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<hudson>
  <disabledAdministrativeMonitors/>
  <version>2.89.3</version>
  <numExecutors>2</numExecutors>
  <mode>NORMAL</mode>
  <useSecurity>false</useSecurity>
  <authorizationStrategy class="hudson.secu
nceLoggedInAuthorizationStrategy">
    <denyAnonymousReadAccess>true</denyAnon
  </authorizationStrategy>
```

Learn to use Jenkins in the right way

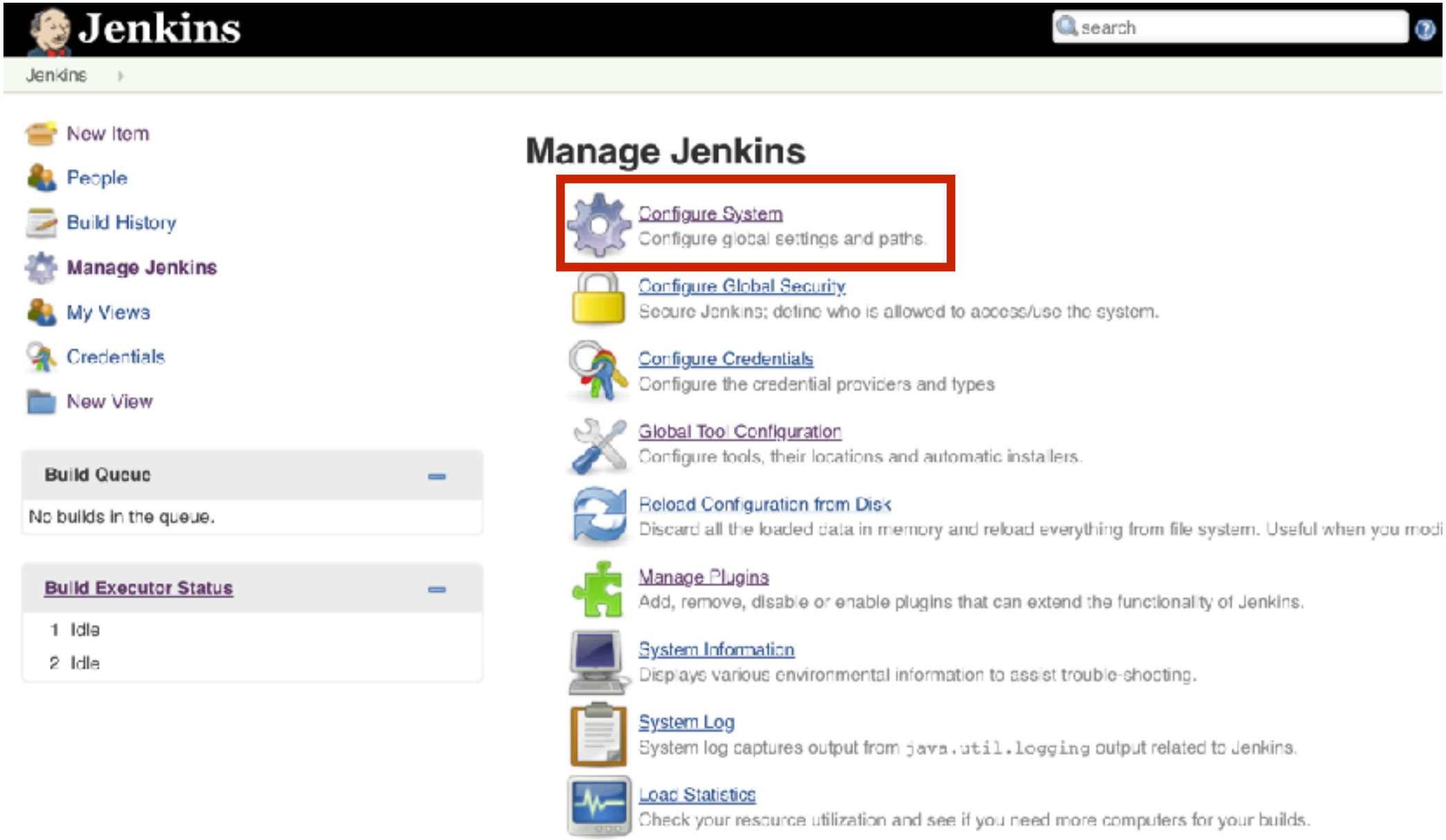
Manage Jenkins

For Administrator to config anything in Jenkins



Configure System

Global setting and paths



Configure System

JENKINS Home

# of executors

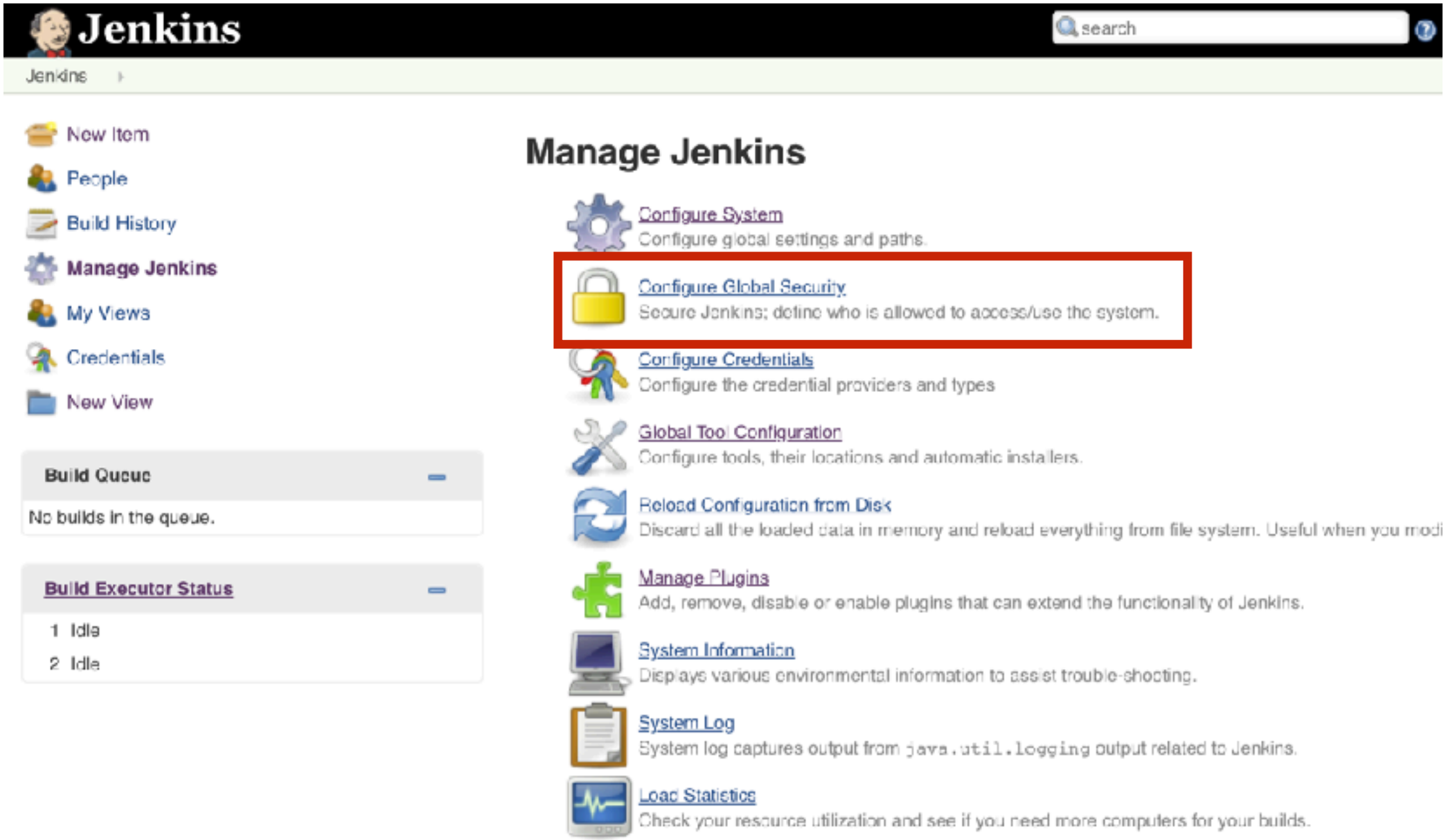
Label name of node

Environment variables

Email notification

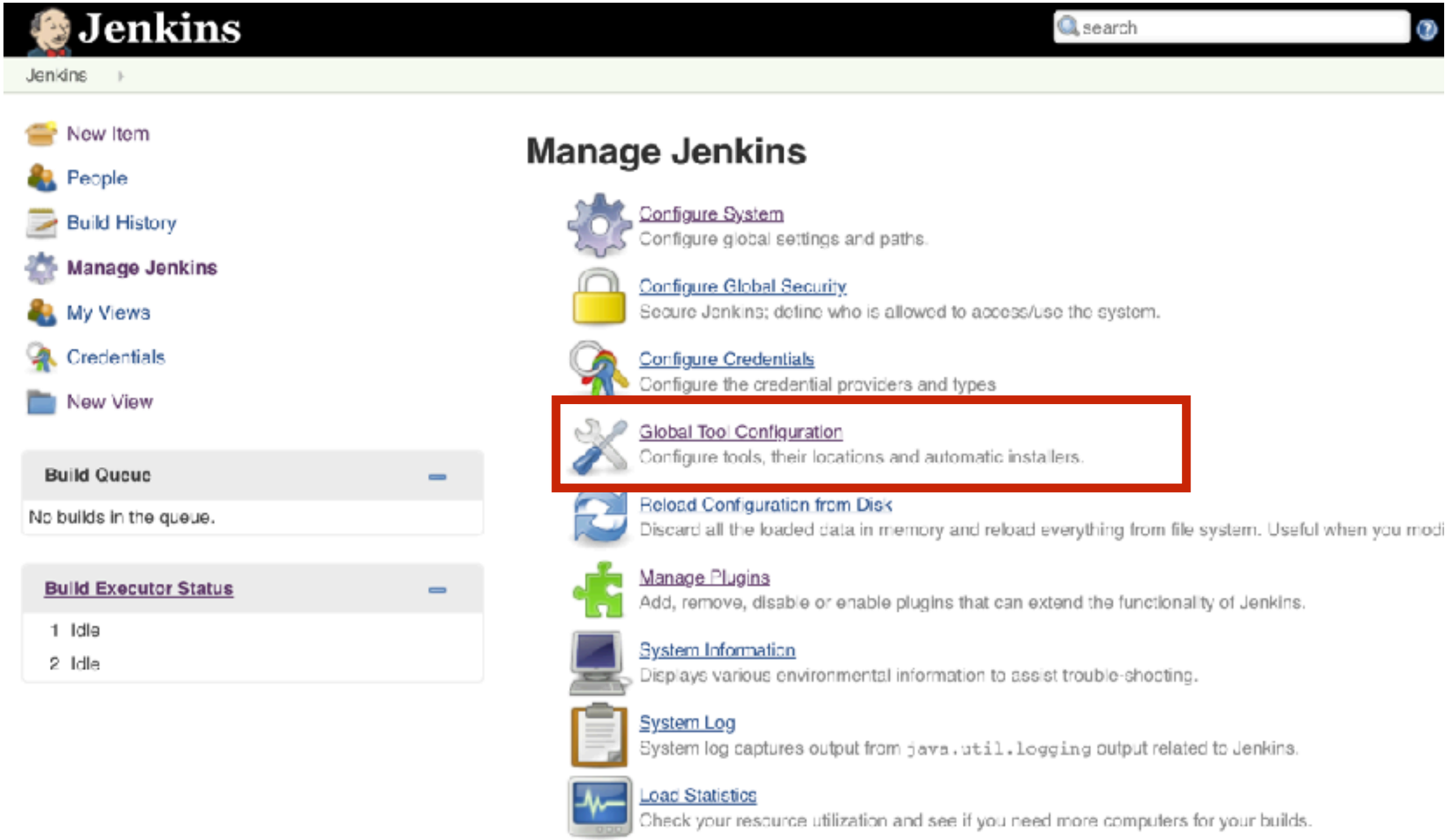
Configure Global Security

Setting for secure Jenkins



Global Tool Configuration

Config tools, location and automatic installers



Global Tool Configuration

Apache Maven

JDK (Java Development Kit)

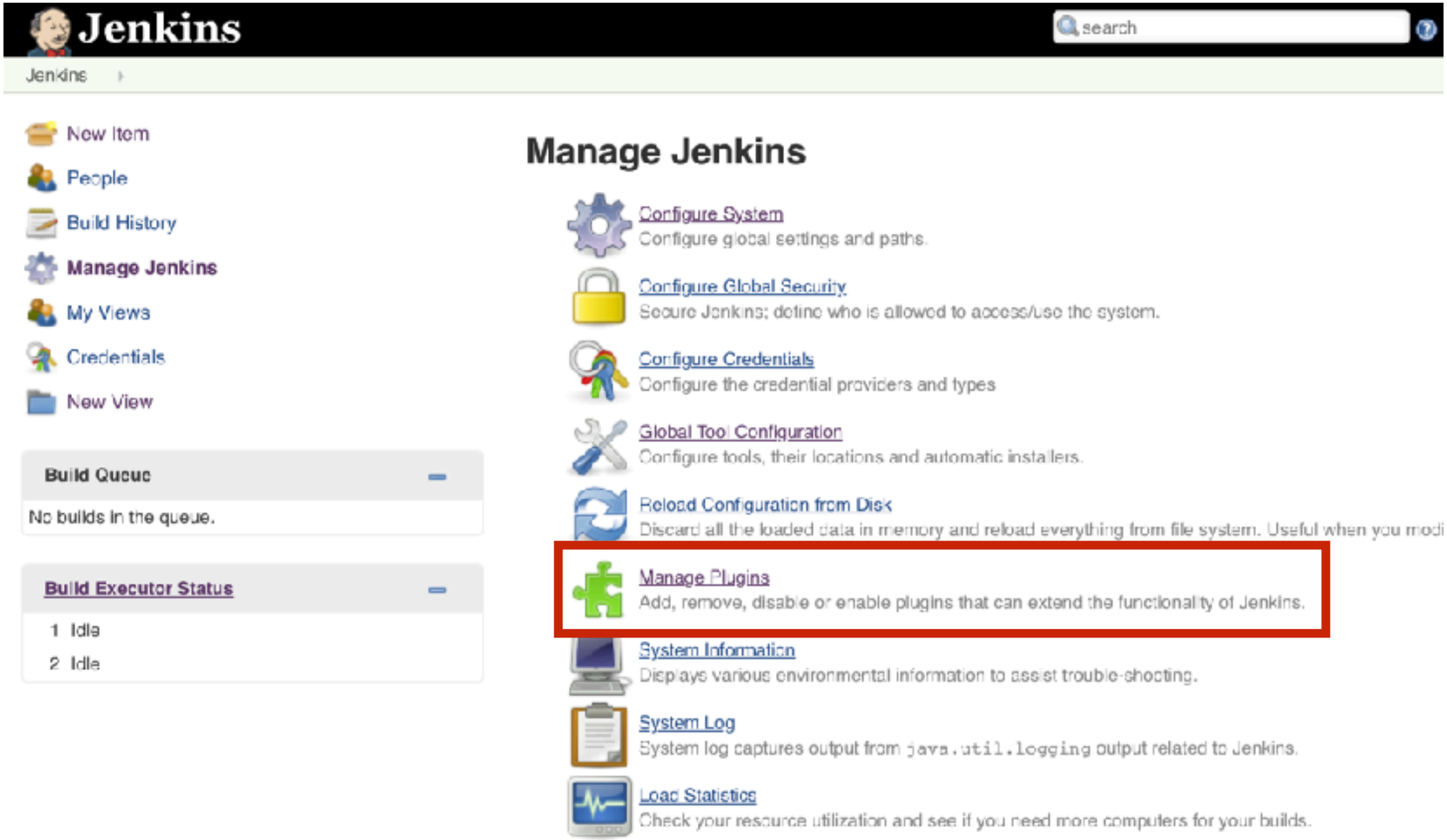
Git

Gradle

Docker

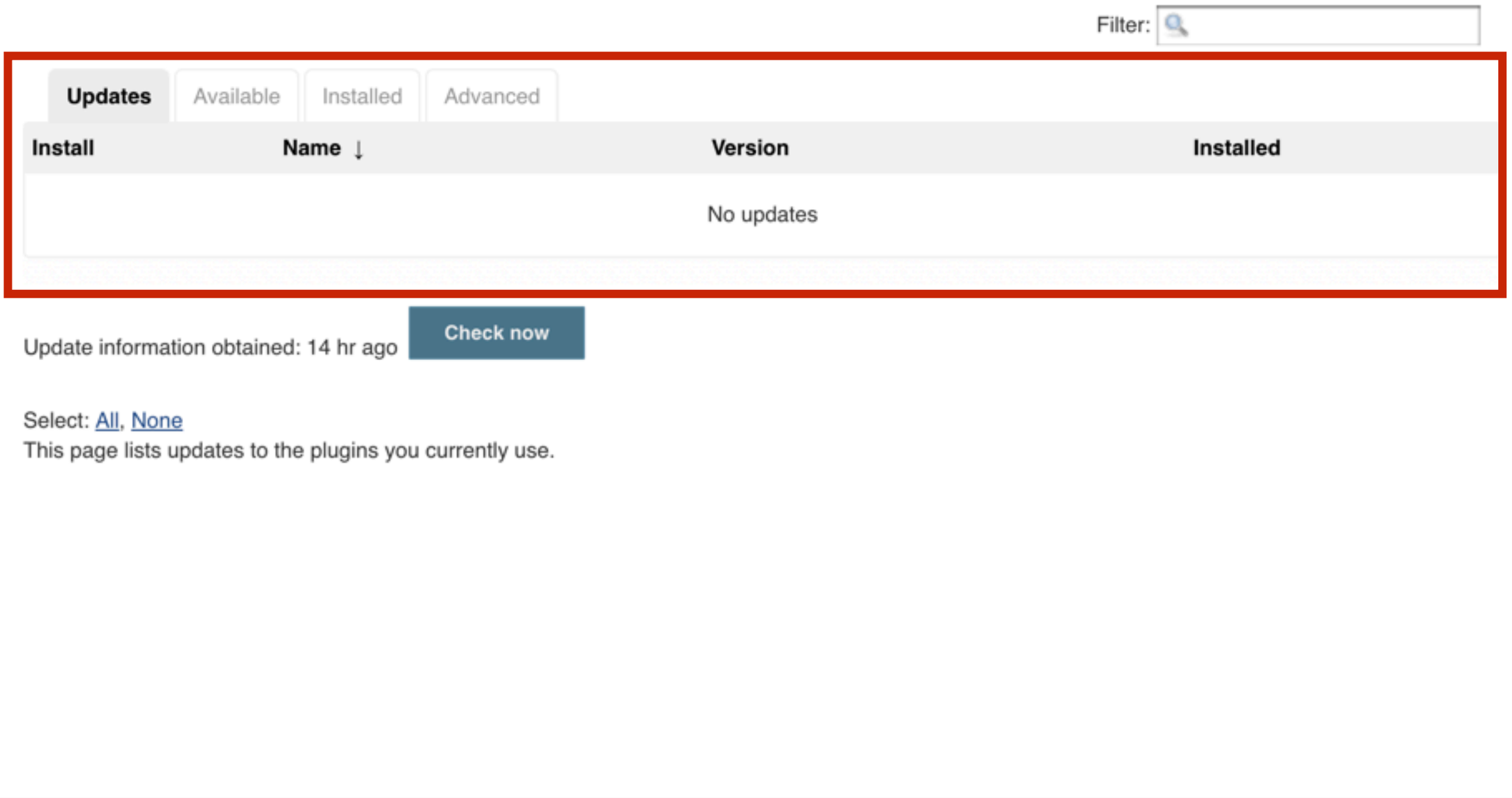
Manage Plugins

Add, remove, enable/disable plugins



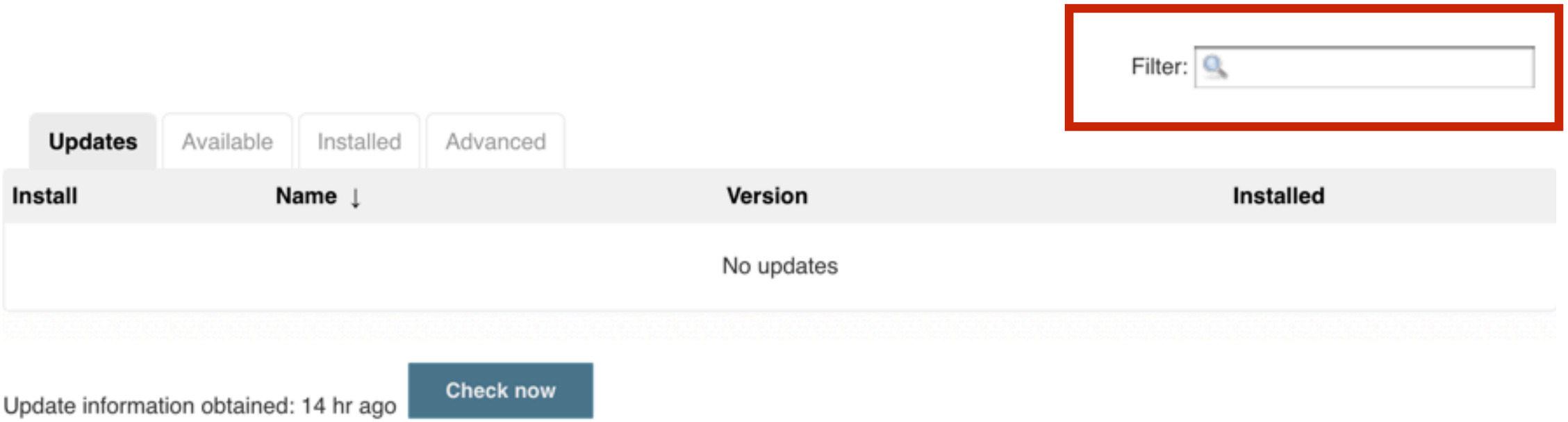
Manage Plugins

Add, remove, enable/disable plugins



Manage Plugins

Filter plugins

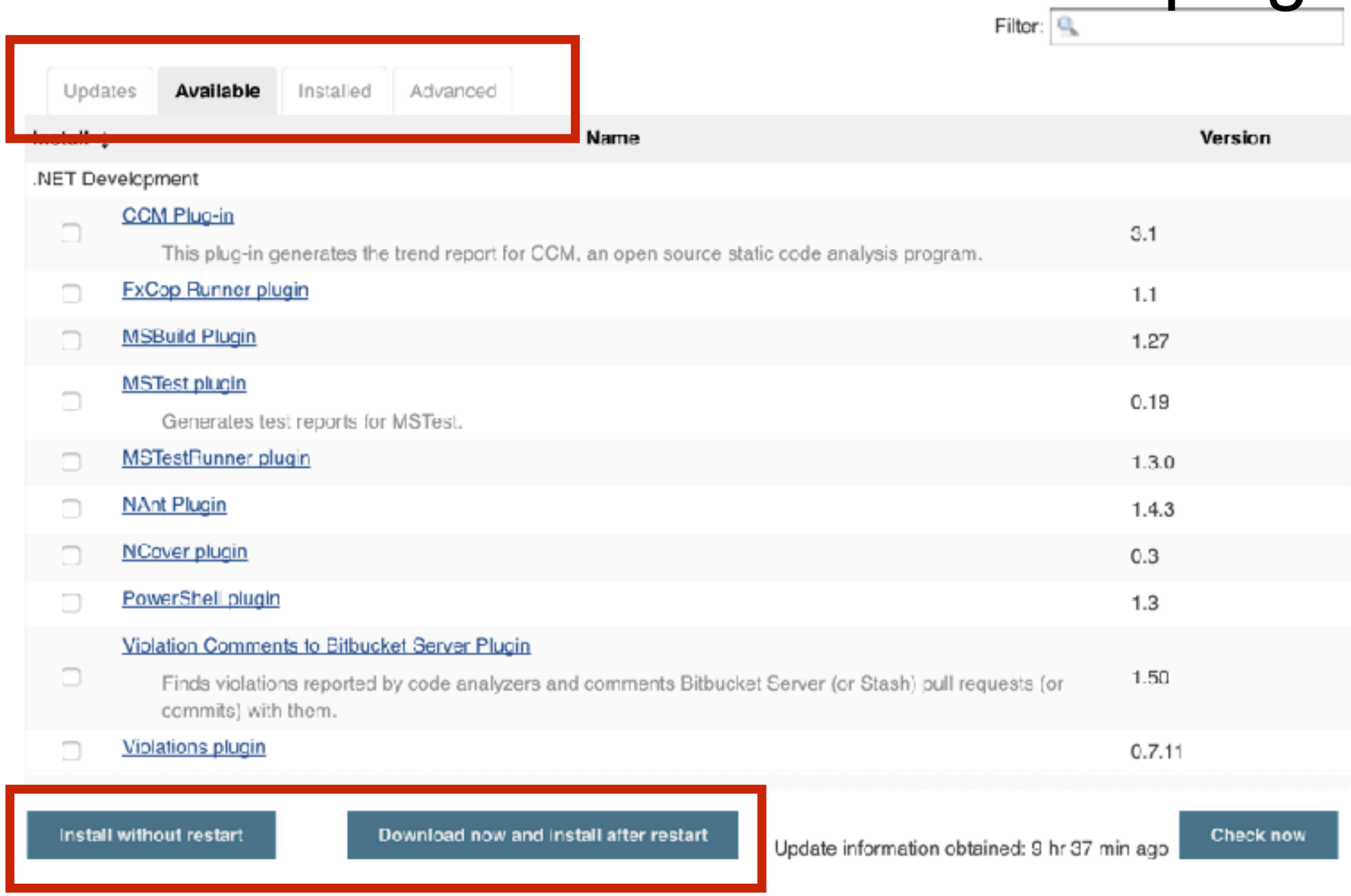


Finds Jenkins's plugin

https://plugins.jenkins.io/

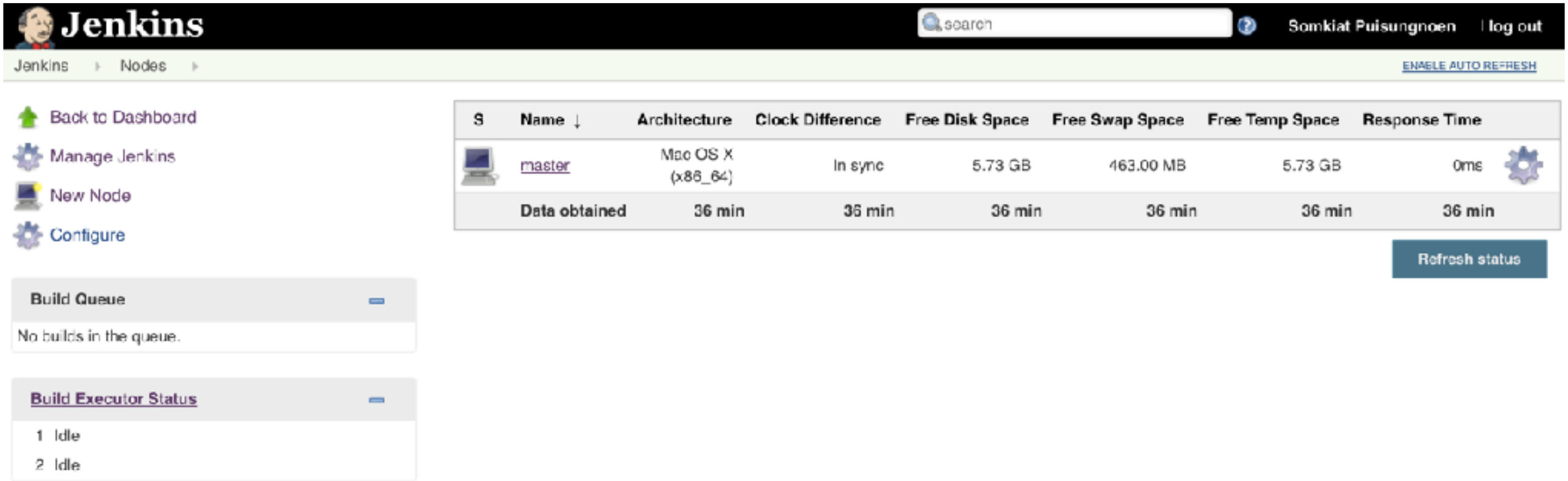
Try to install a first plugin

Choose Available tab and select a plugin



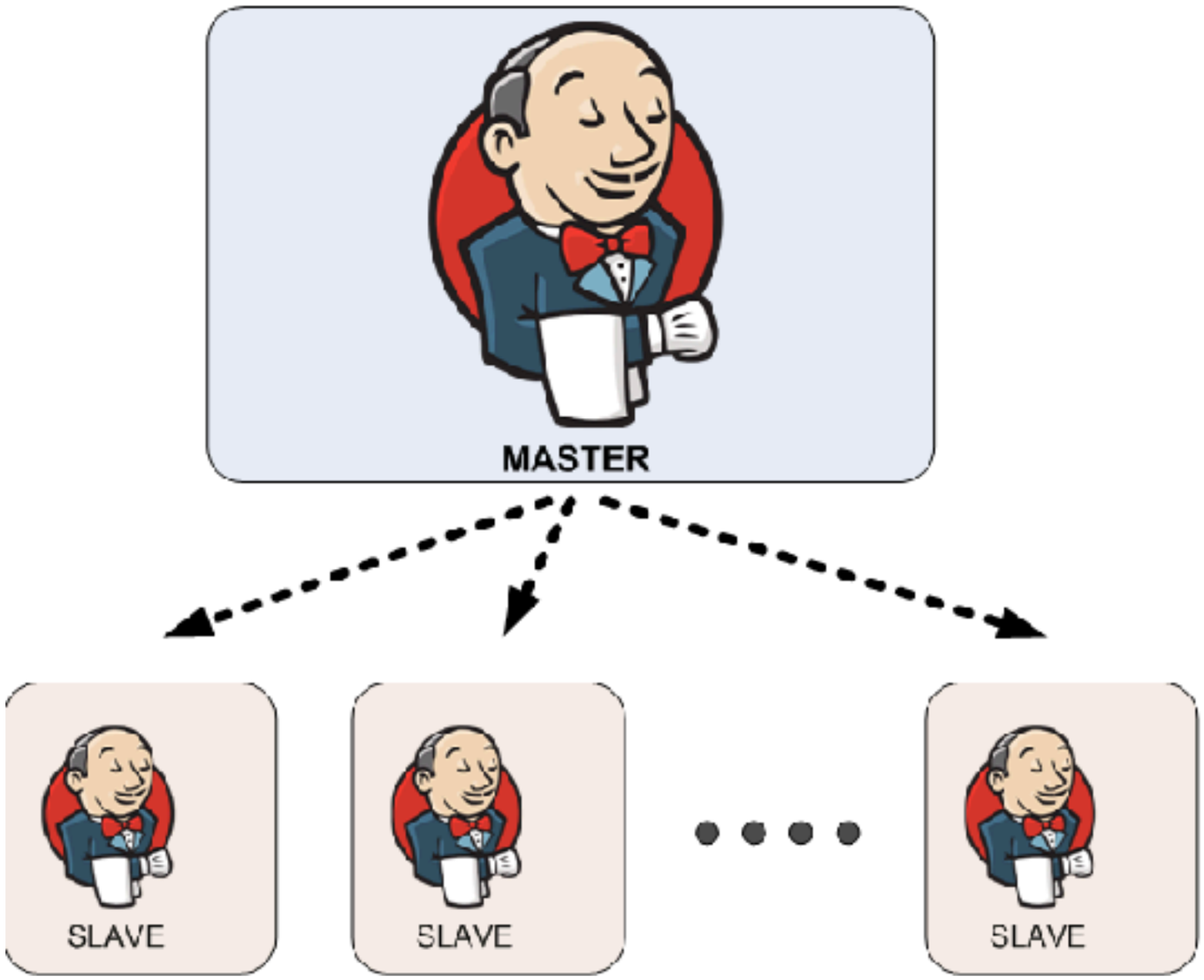
Manage Nodes

Add, remove, status of nodes



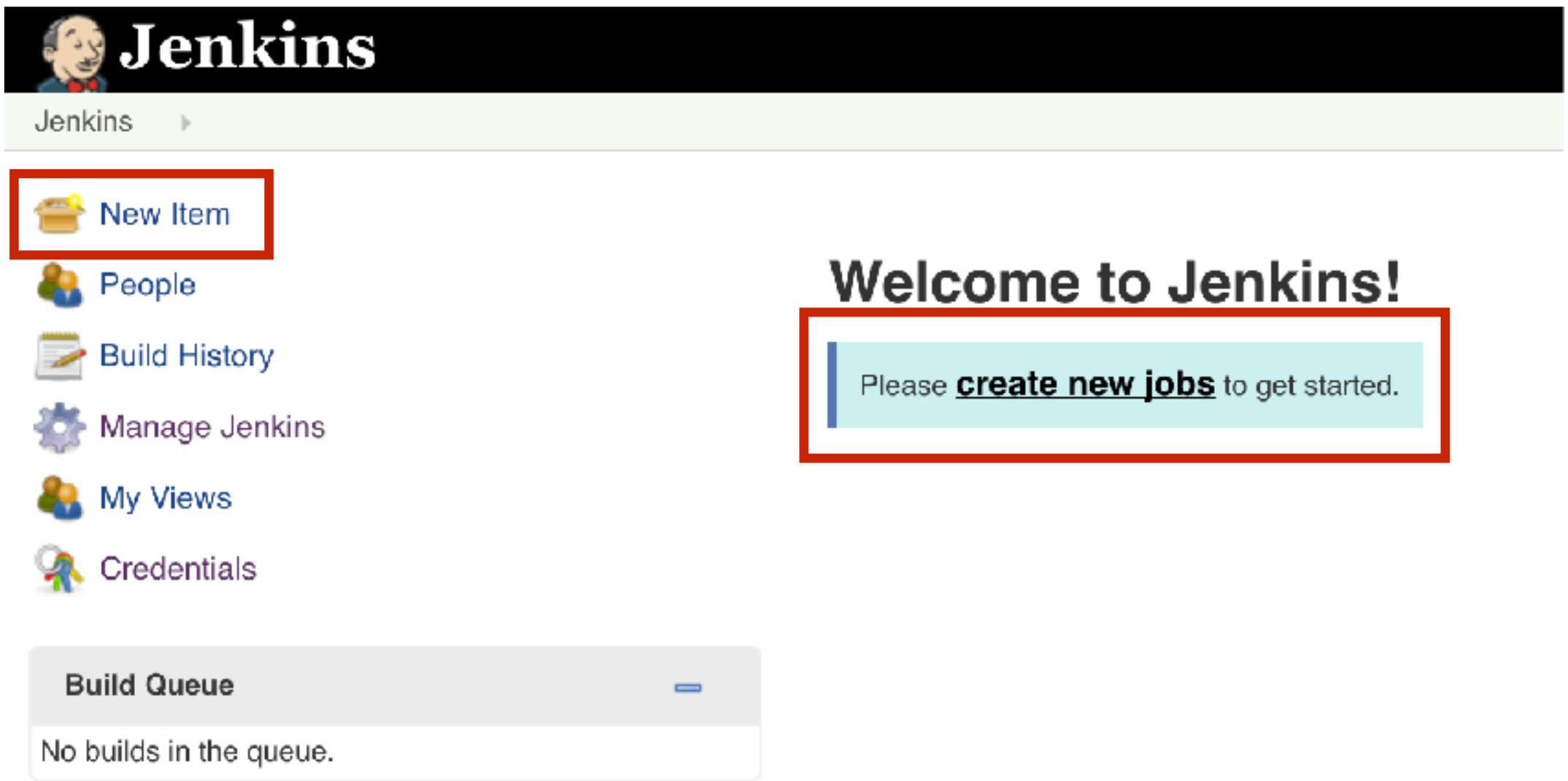
Manage Nodes

Master-slave concept to scale Jenkins

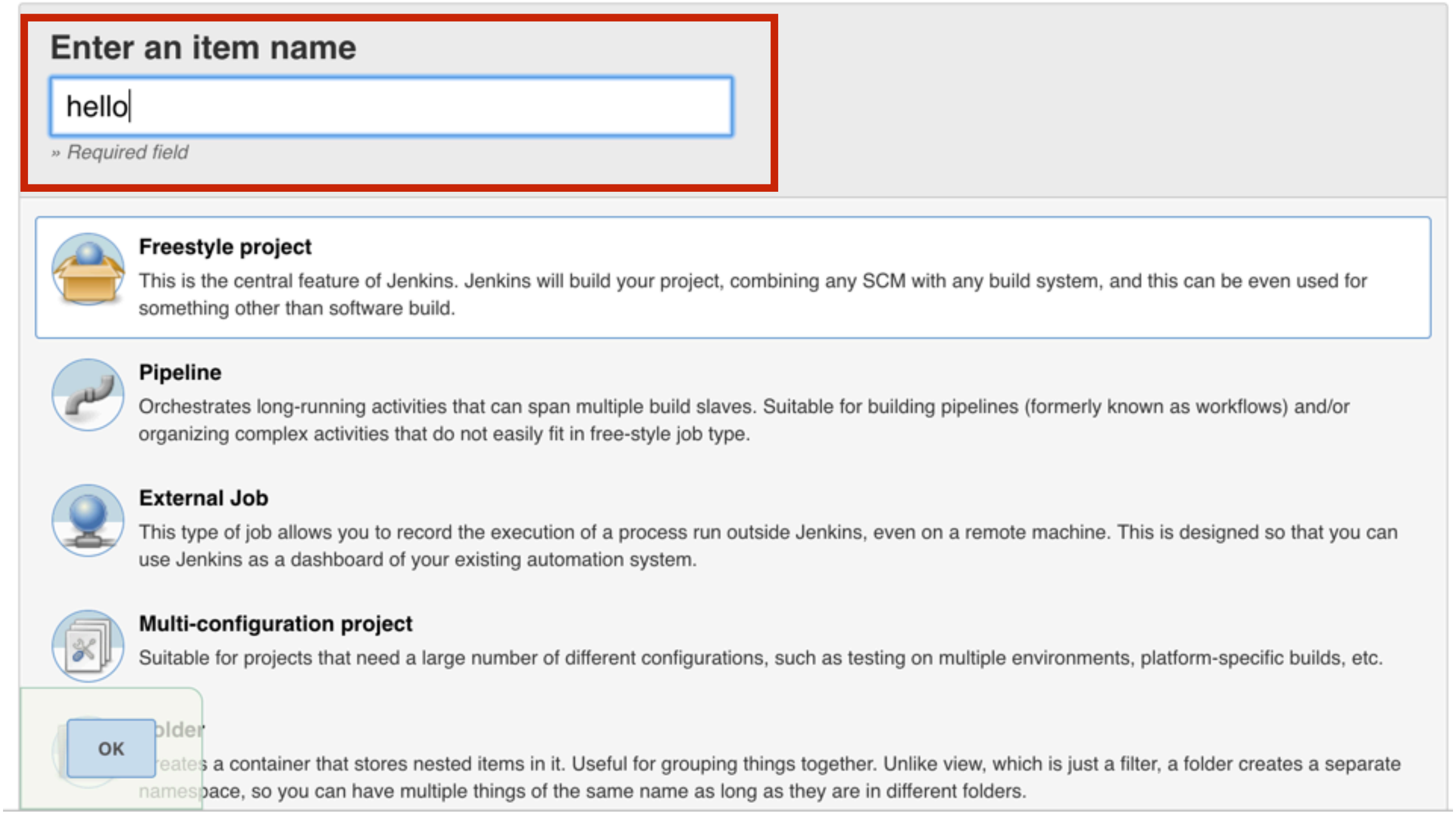


Create a first Job

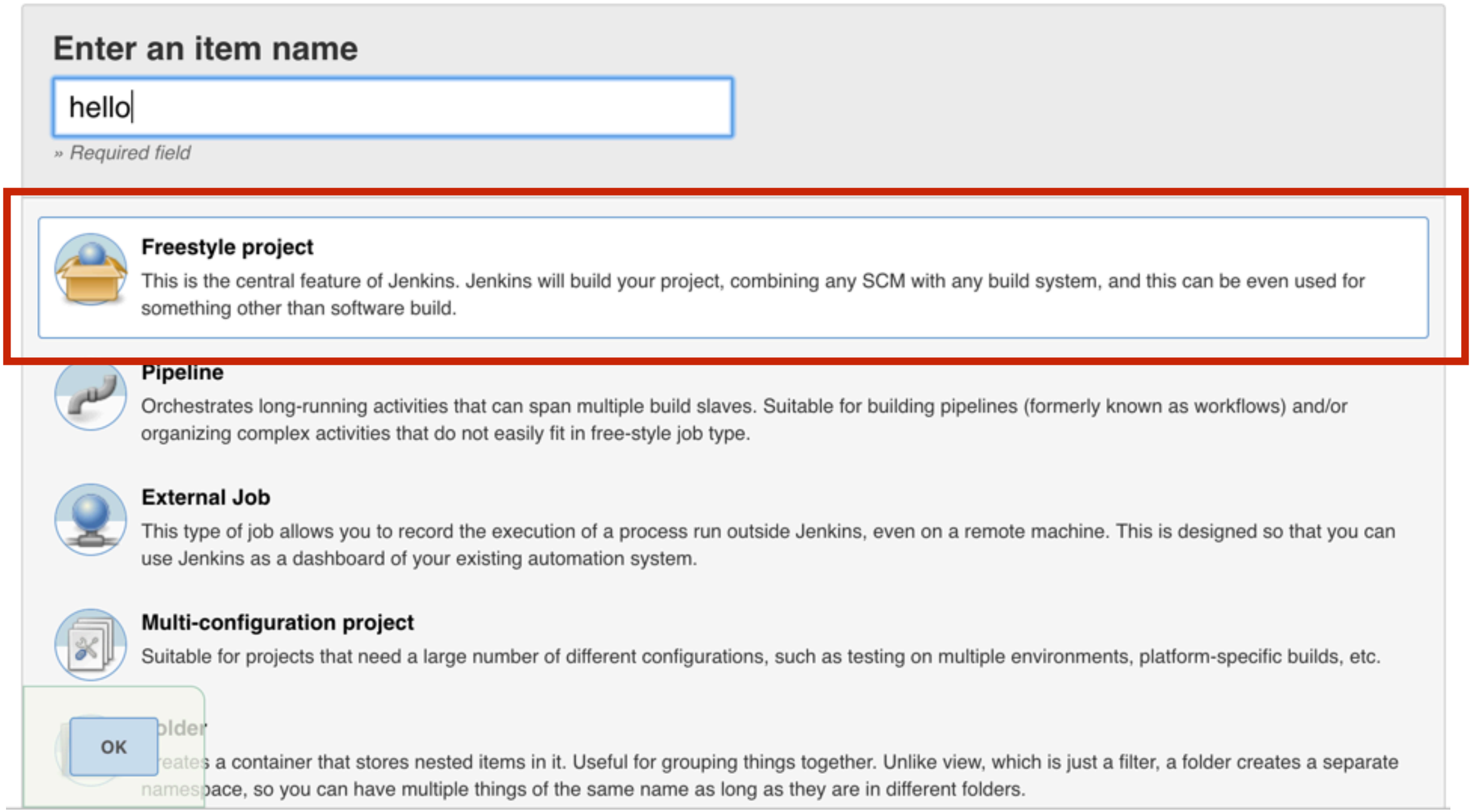
1. Create a new job



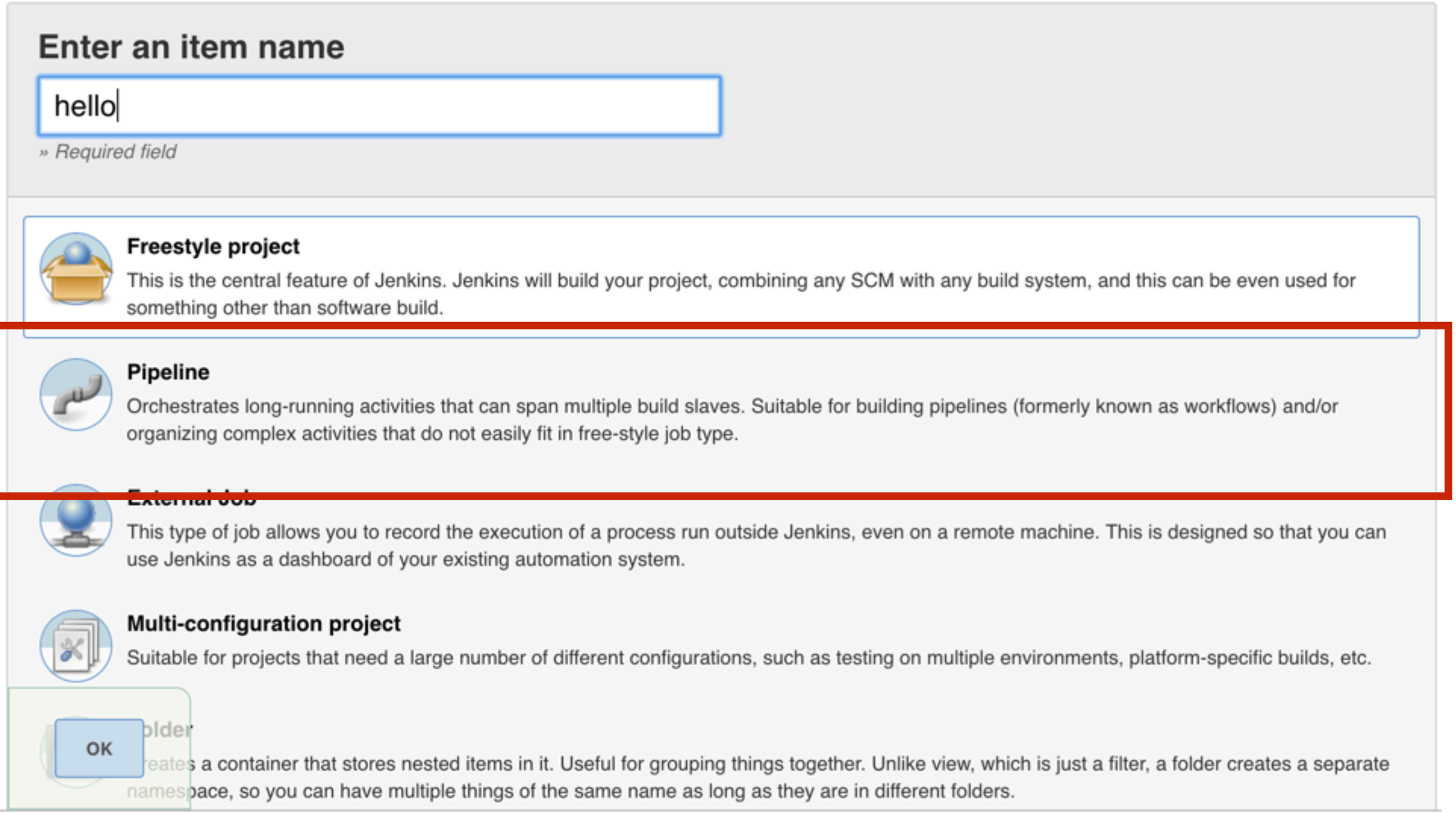
2. Fill in a job name



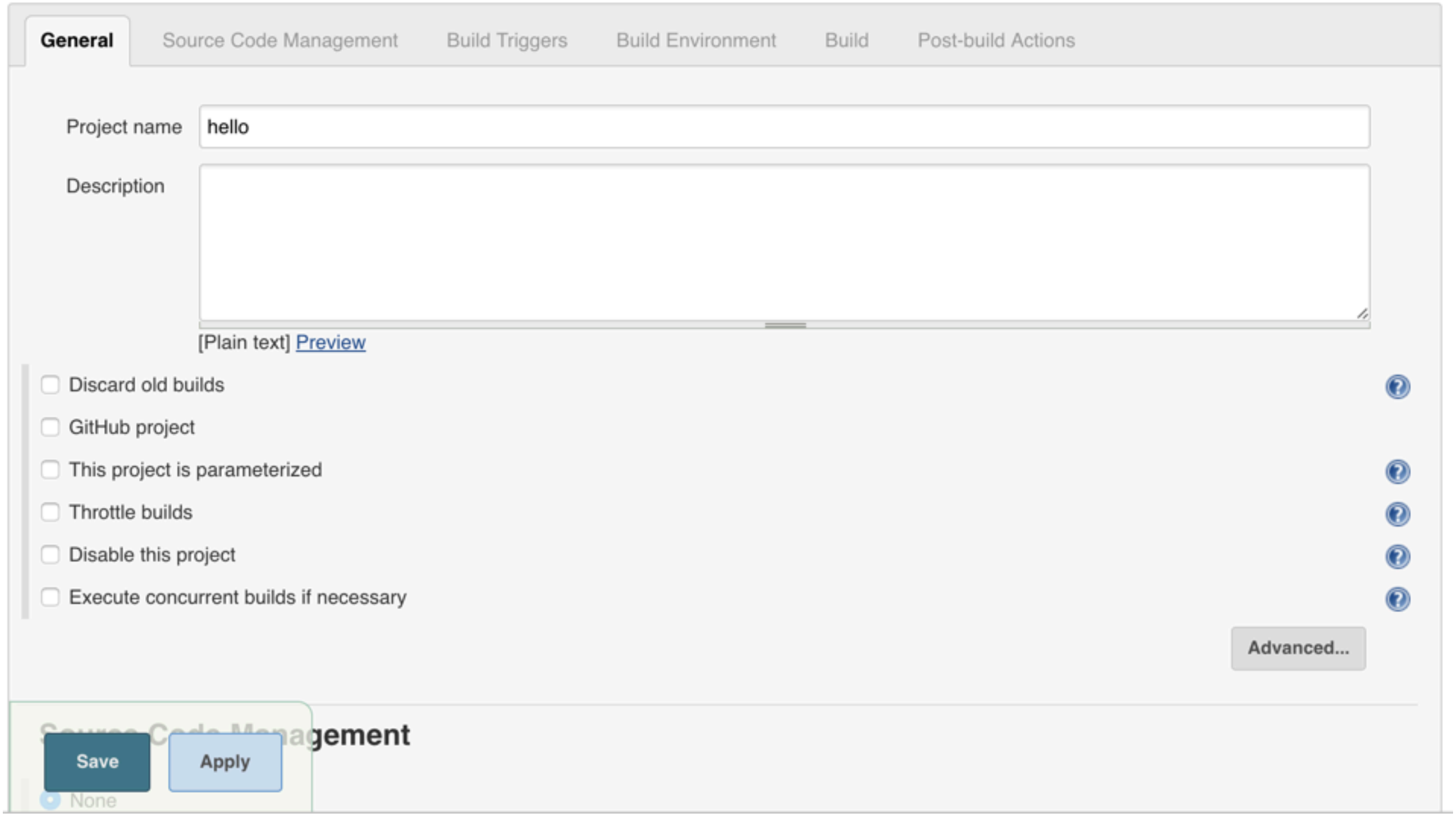
3. Choose type of job



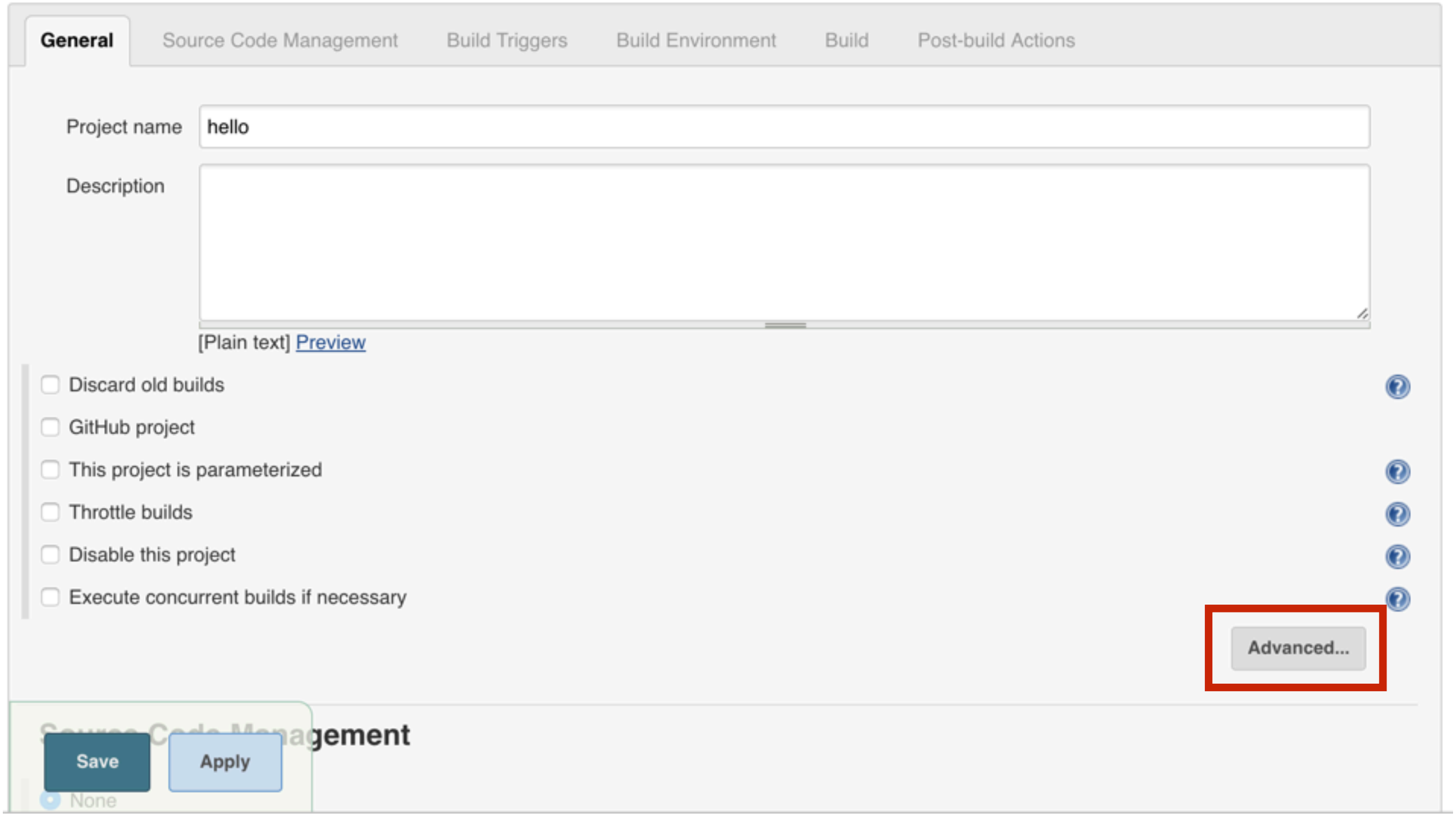
3. Choose type of job



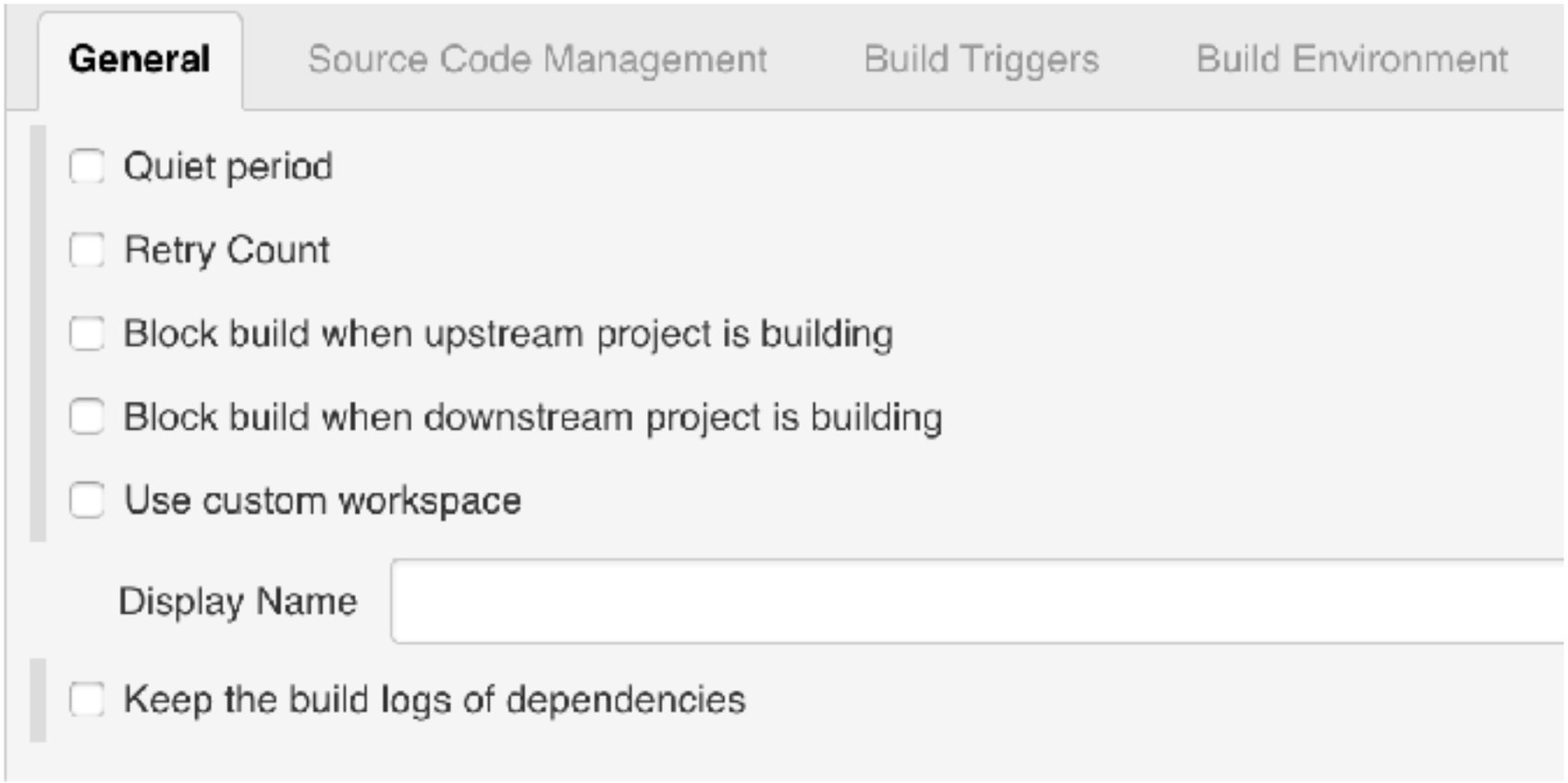
4. General section



4.1 Advanced options

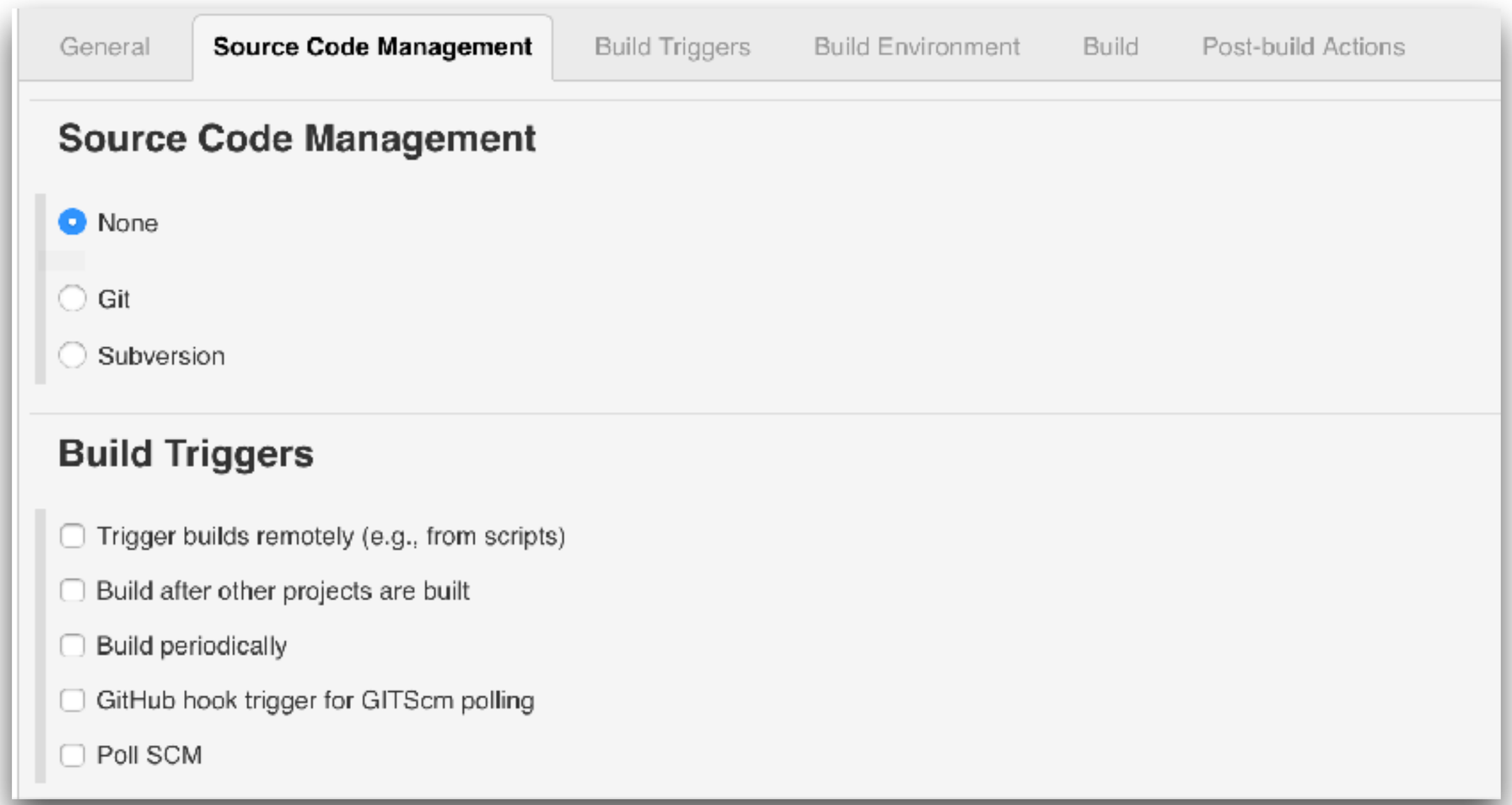


4.1 Advanced options



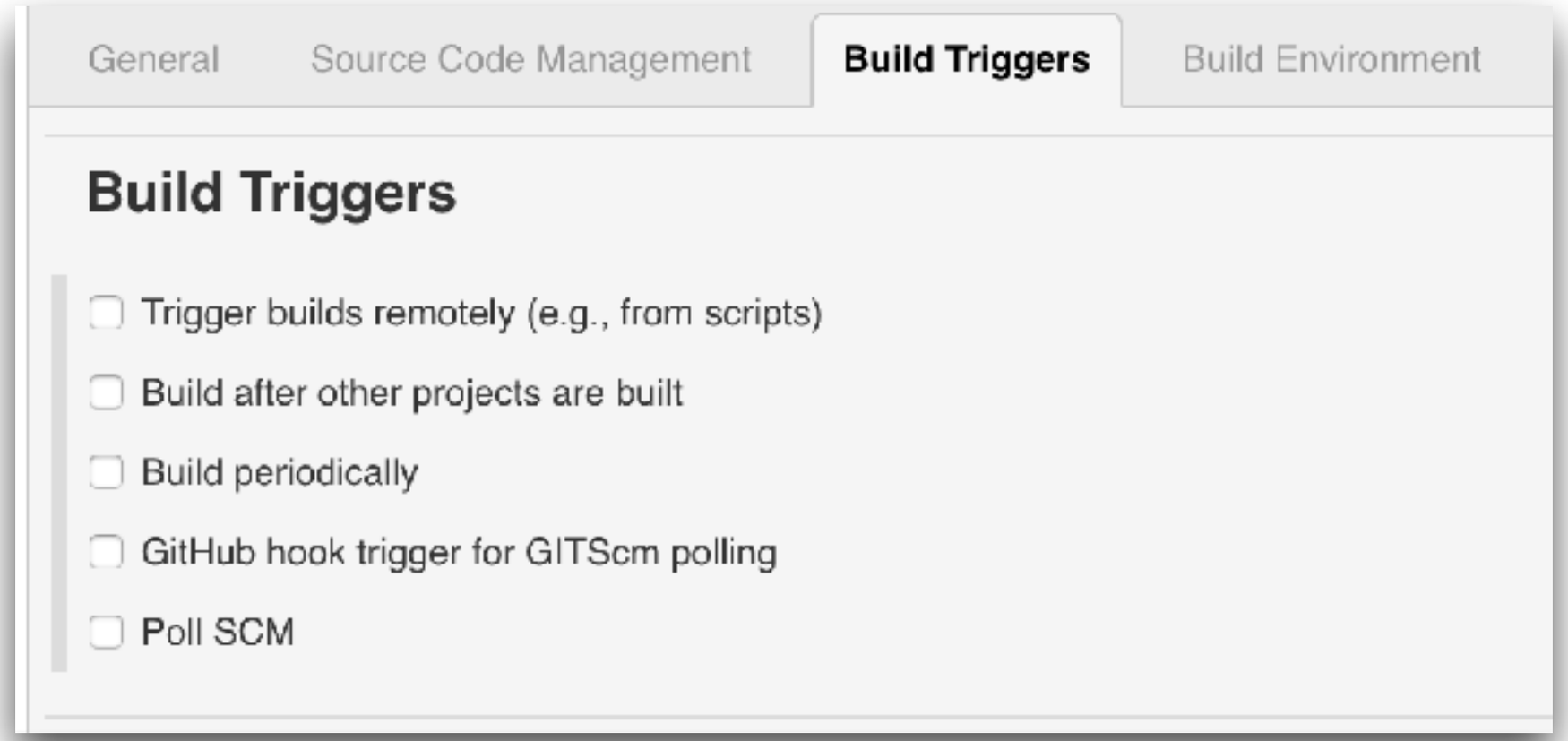
5. Source code management

By default is Git and Subversion

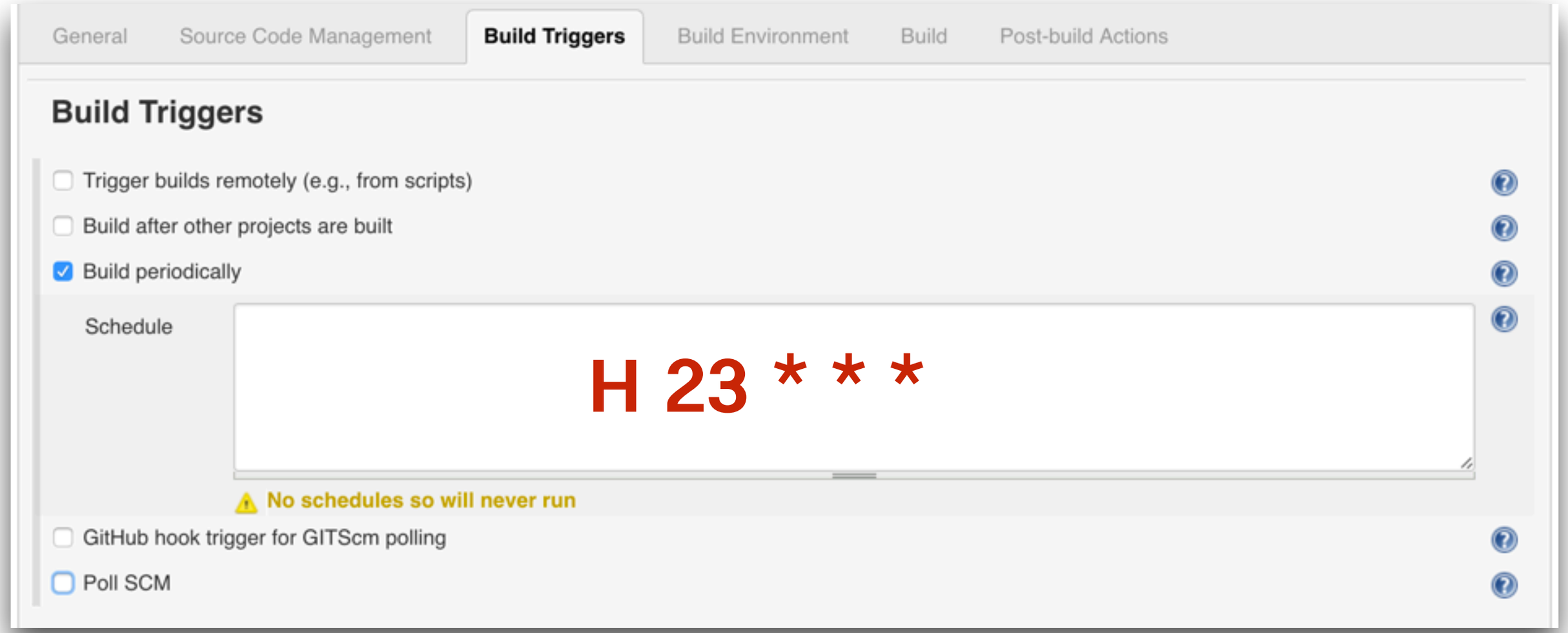


6. Build trigger

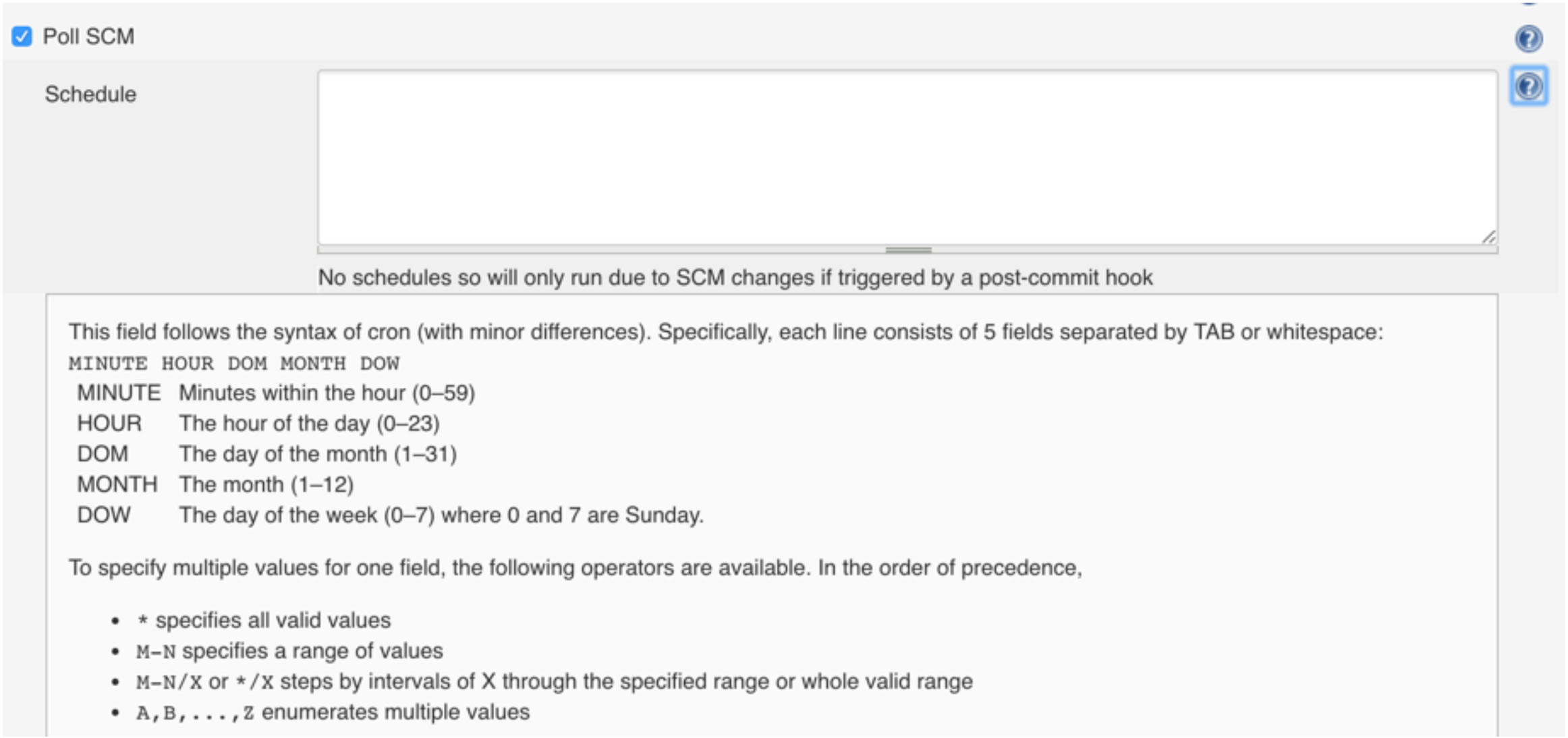
When to run this job



6.1 Periodically

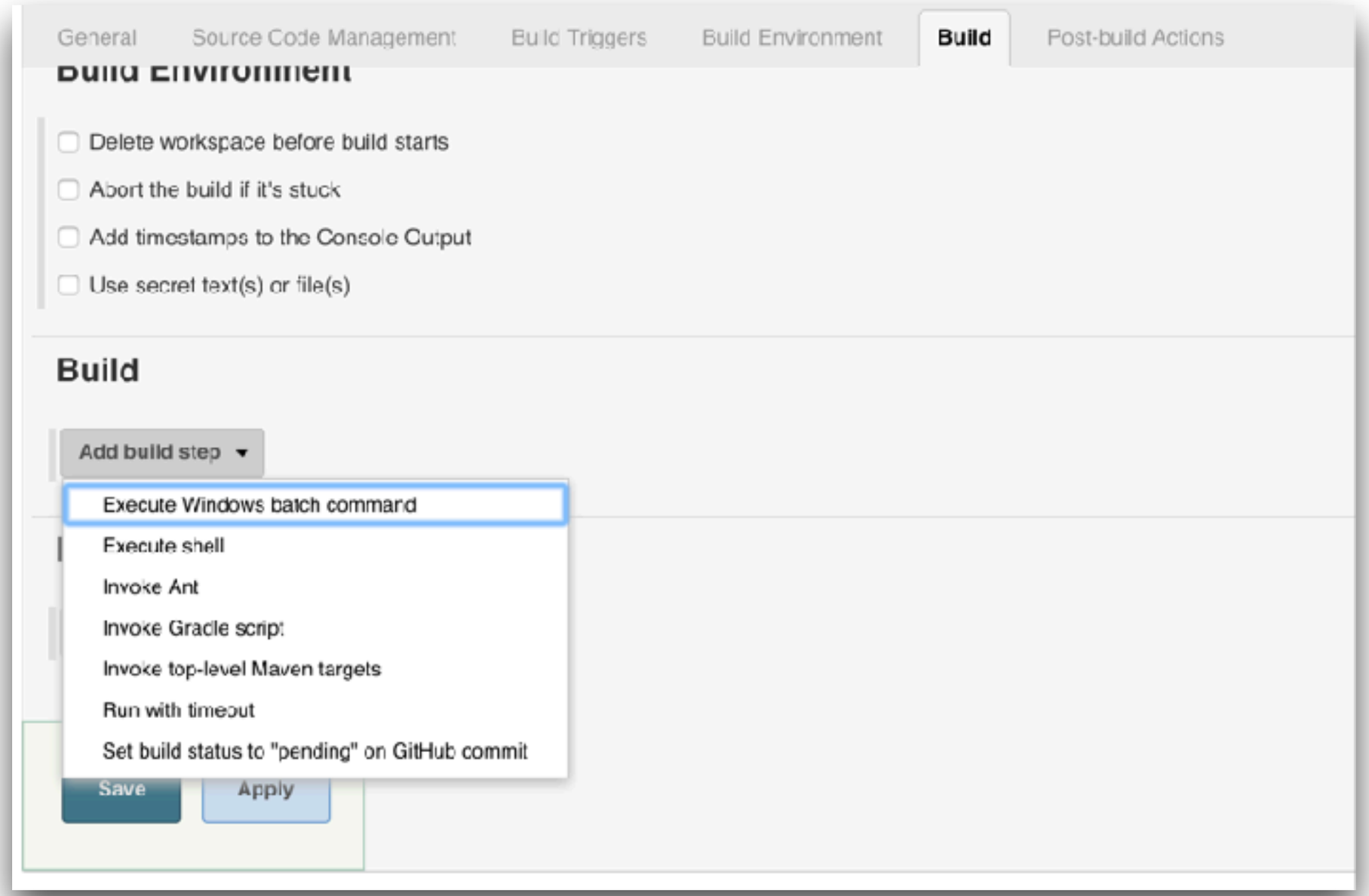


6.2 Poll SCM

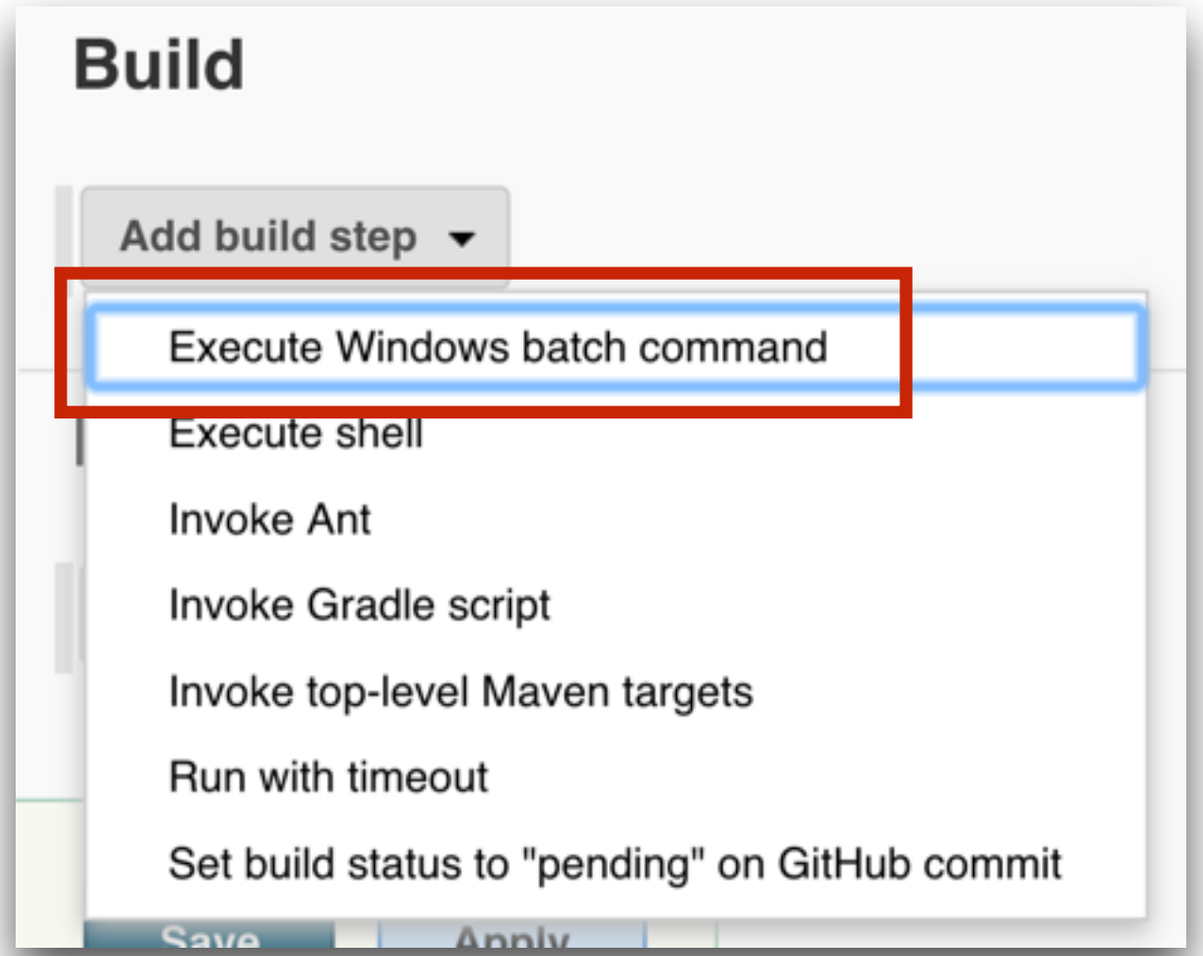


7. Add build step

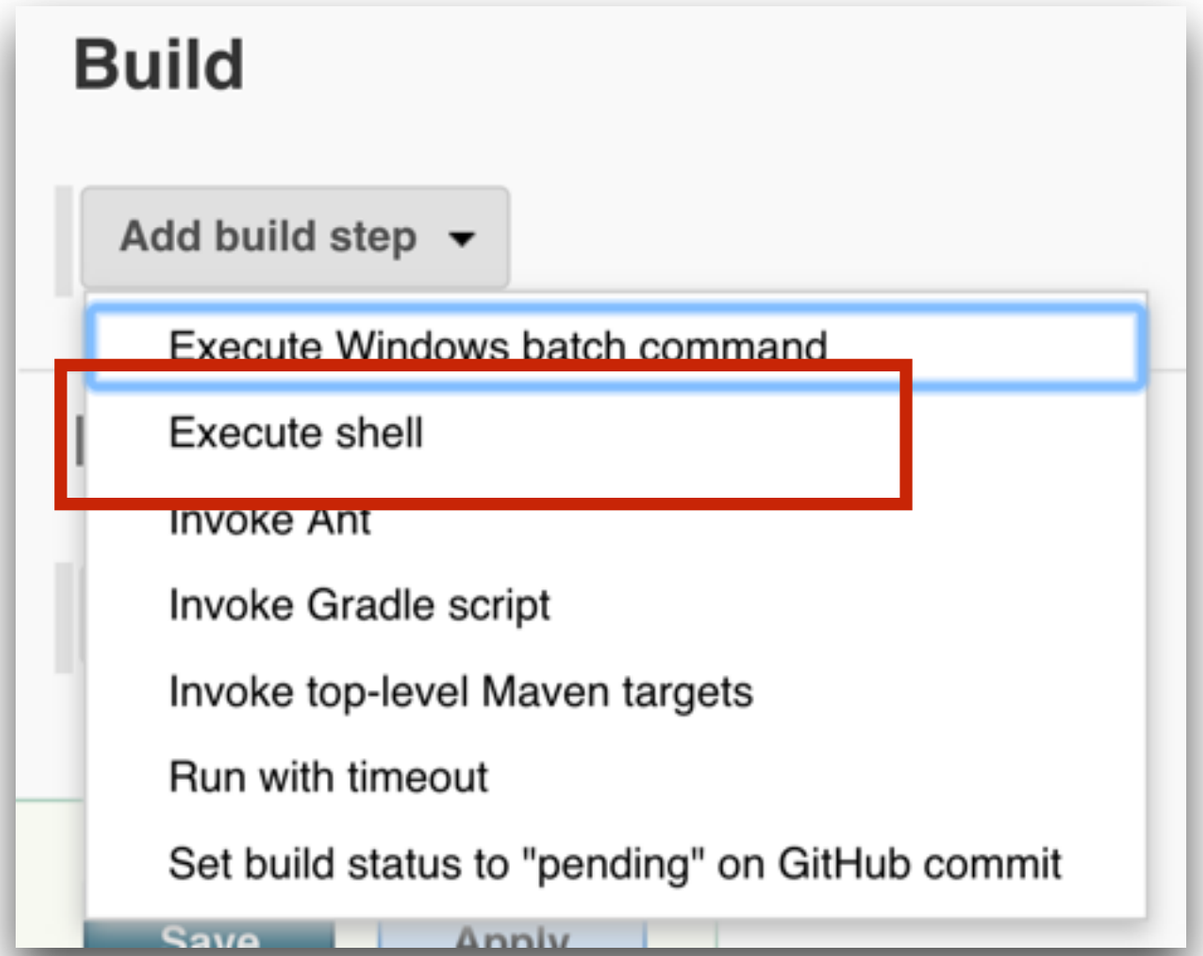
What to run this job



7.1 For Windows



7.2 For Linux/Mac



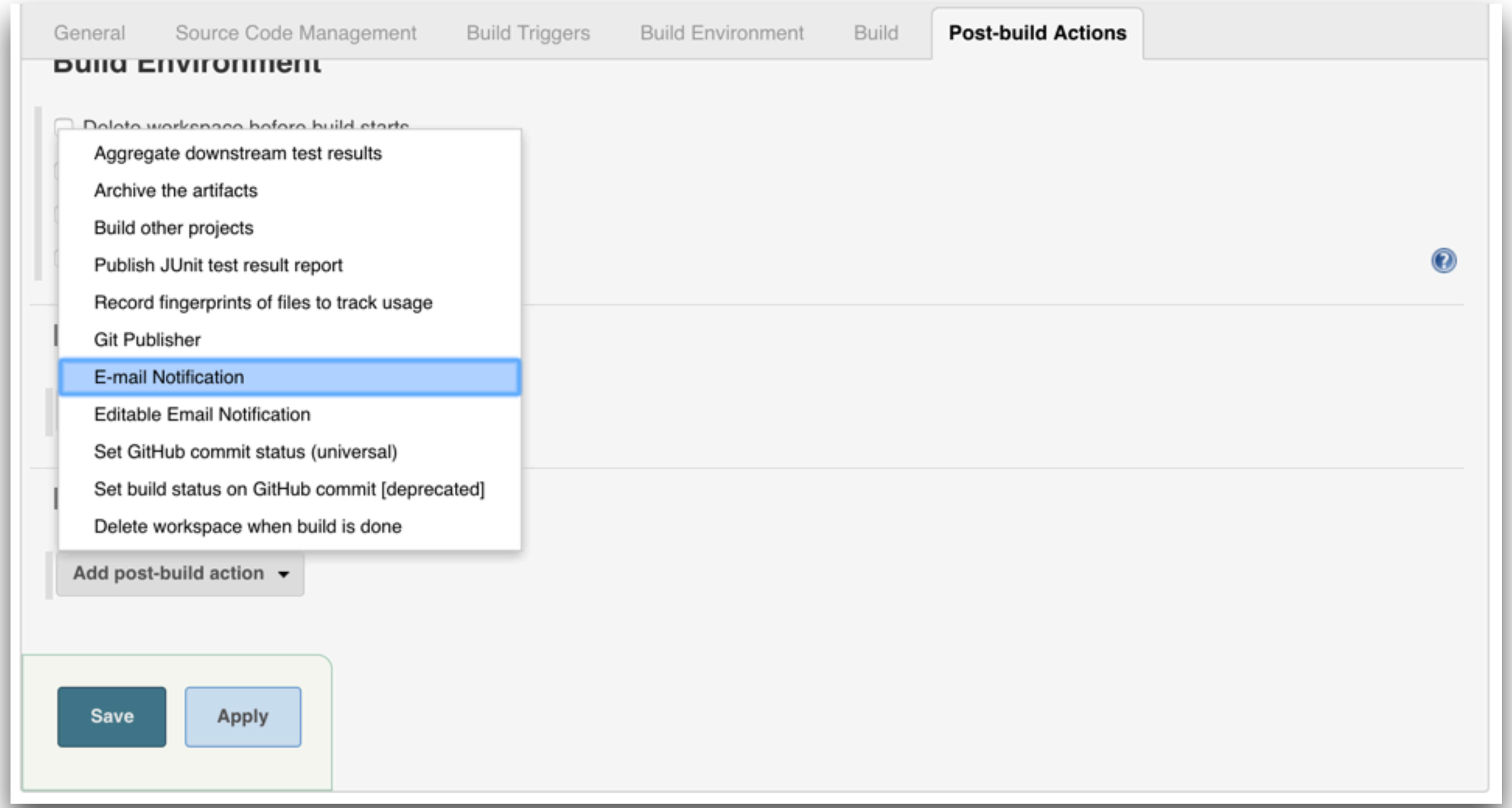
8. Post build actions

Generate reports

Send email

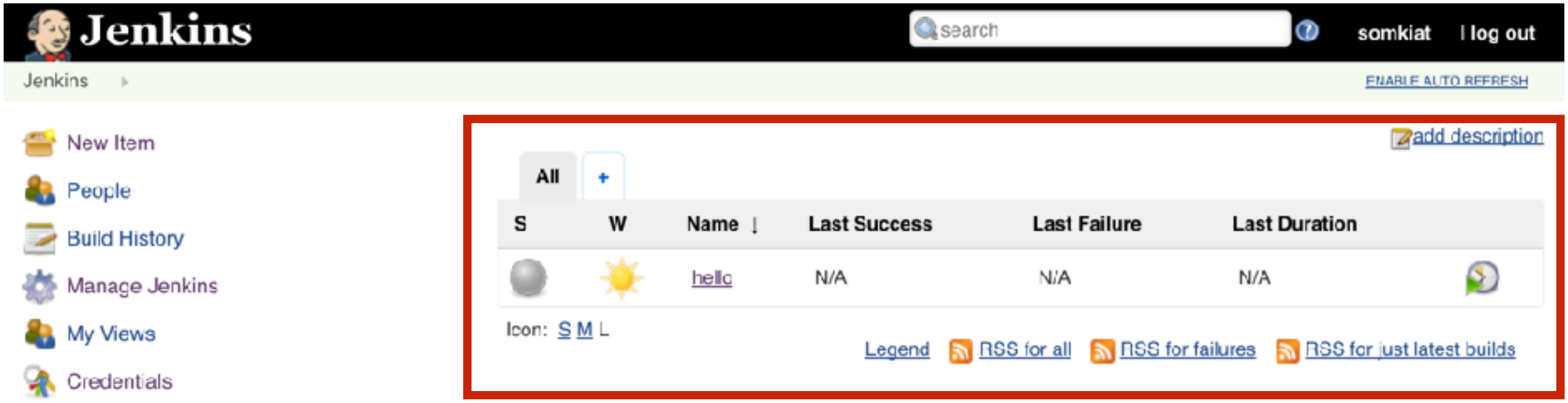
Run other jobs/projects

8. Post build actions



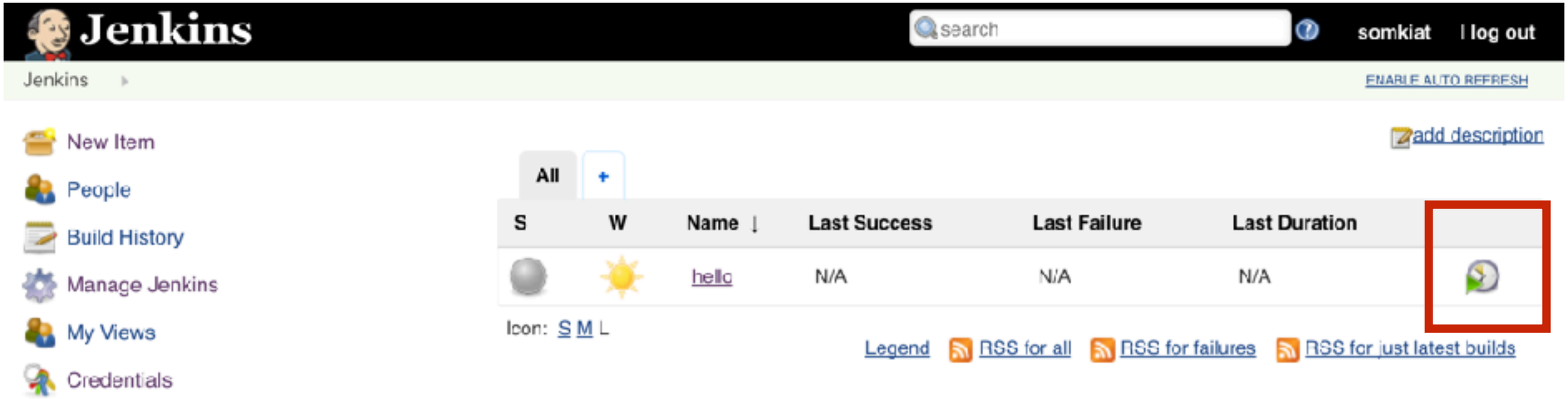
9. Run your job

Manual and Scheduler run



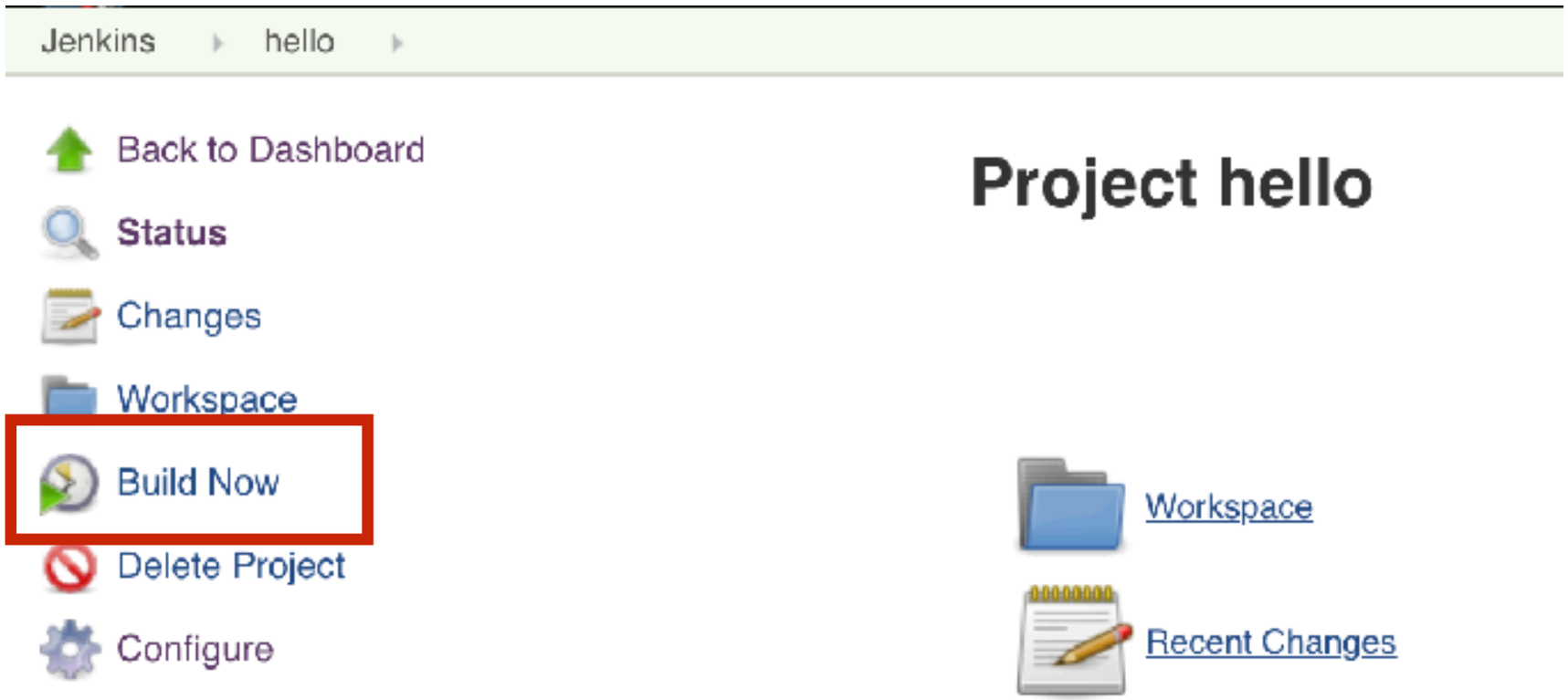
9. Run your job

Start build your job



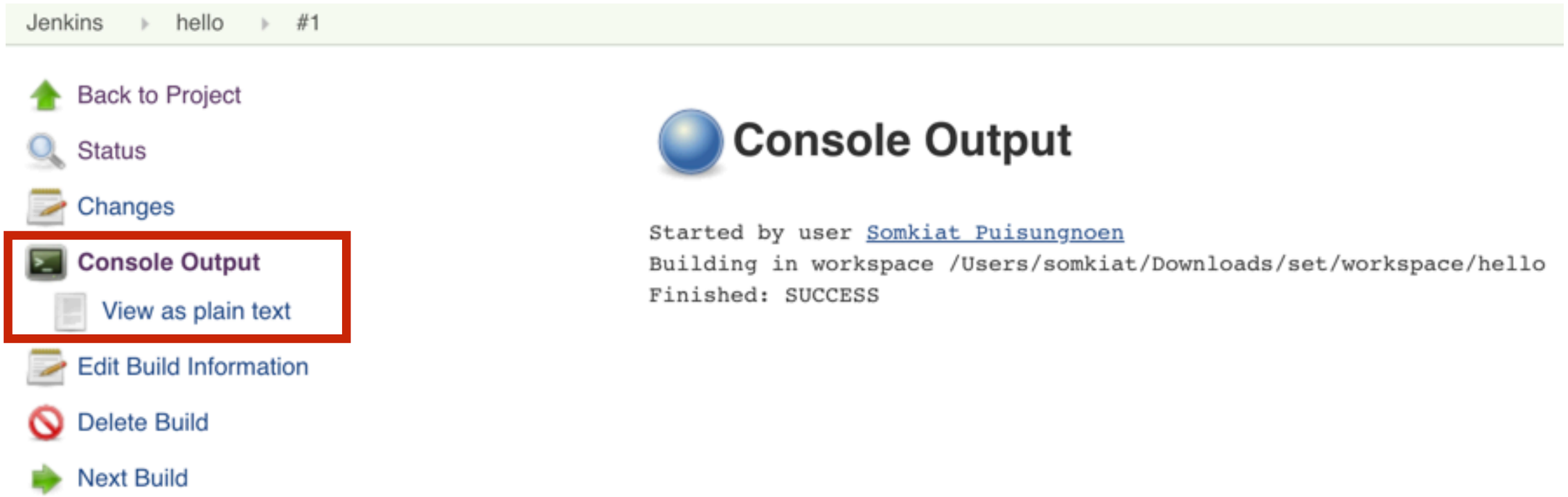
9. Run your job

Start build your job



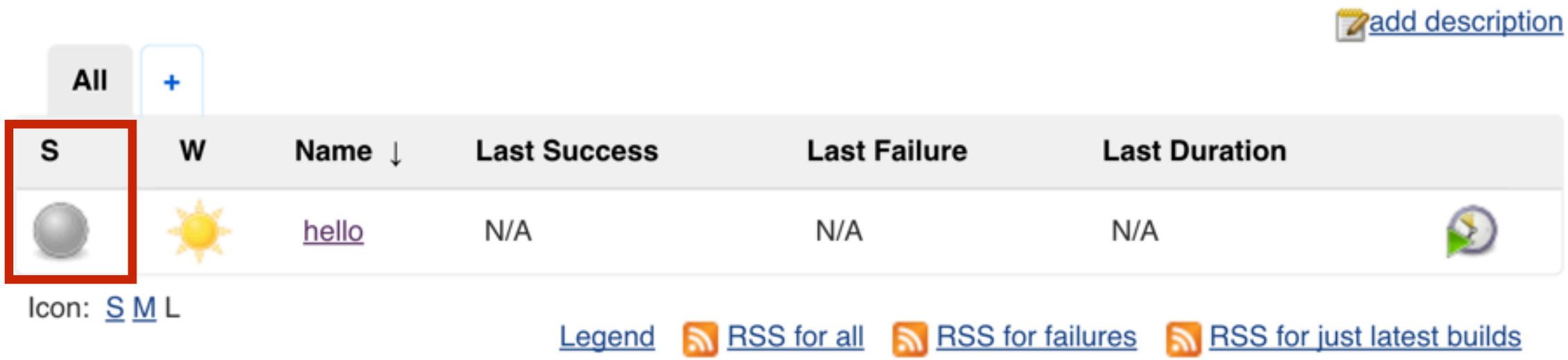
9. Run your job

See your job's output



10. See job's status

By default is Blue, Red and Gray

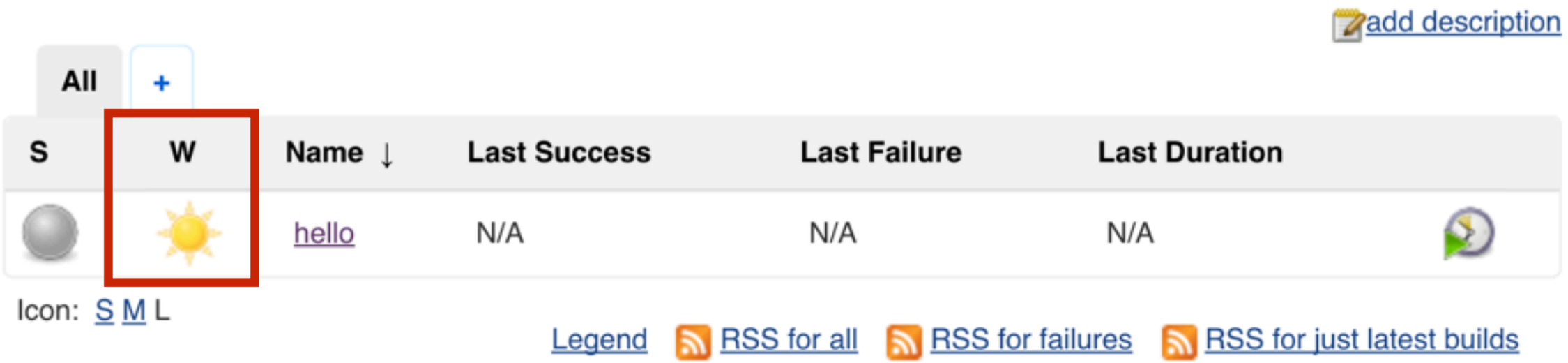


Blue = build success

Red = build failure

Gray = disabled/never executed

11. See job's health



Sunny = 100% success rate
Cloudy = 60% success rate
Raining = 40% success rate

Let’s workshop