How to Become a Software Developer

From Scratch

Introduction

How does it work
in general?

Module 1
IT Industry

How does it look
like daily?

Module 2
Daily Work

What are you
actually building?

Module 3
The Project

Enterprise Web Application

IT Industry

Module 1
IT Industry

Module 2
Daily Work

Module 3: The Project

Module 4: Basics of Software Development

Java | Spring | Frontend Development | Version Control Systems | Databases | Network and Infrastructure

Module 5: Advanced Software Development

Principles | Patterns | Techniques

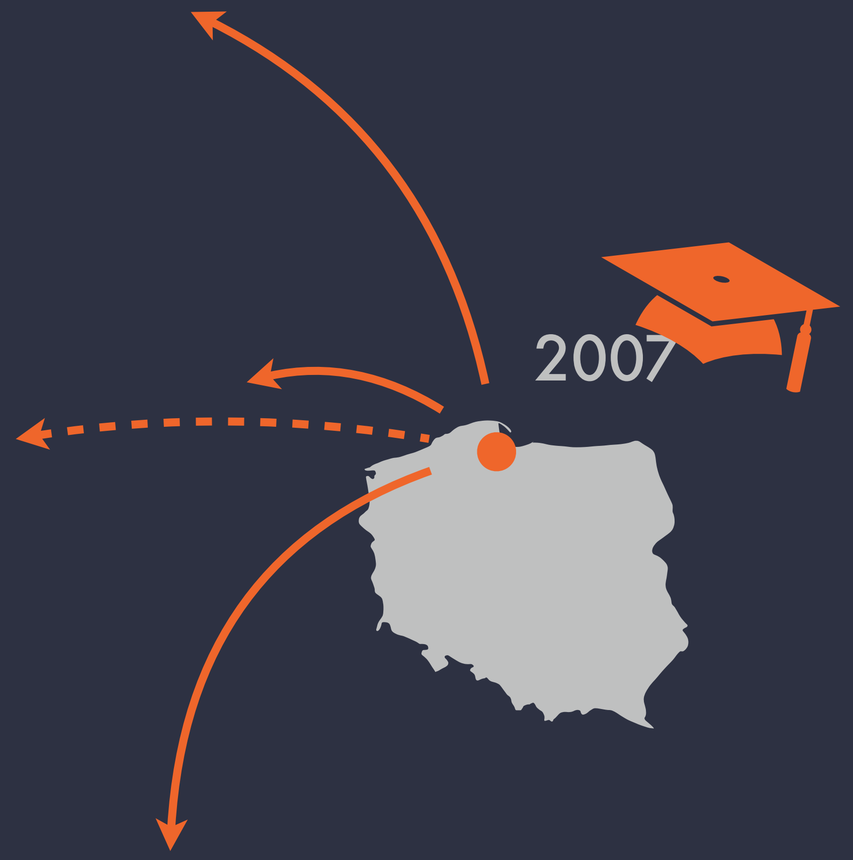
Module 6: Practicalities

Job Interviews | First Days | Learning Tips

IT Industry

- Java
- Spring
- JavaScript
- Angular
- PostreSQL
- Docker
- Git
- Mid-class PC with Windows
- IntelliJ IDEA
- Visual Studio Code

Bartosz Kaminski



- Clean code
- Design and architecture
- Solving problems

Module 1

IT Industry

What Is It All About?

Passion

Passion

Software Development ⟷ Problem Solving

Types of software

Embedded software

Operating systems

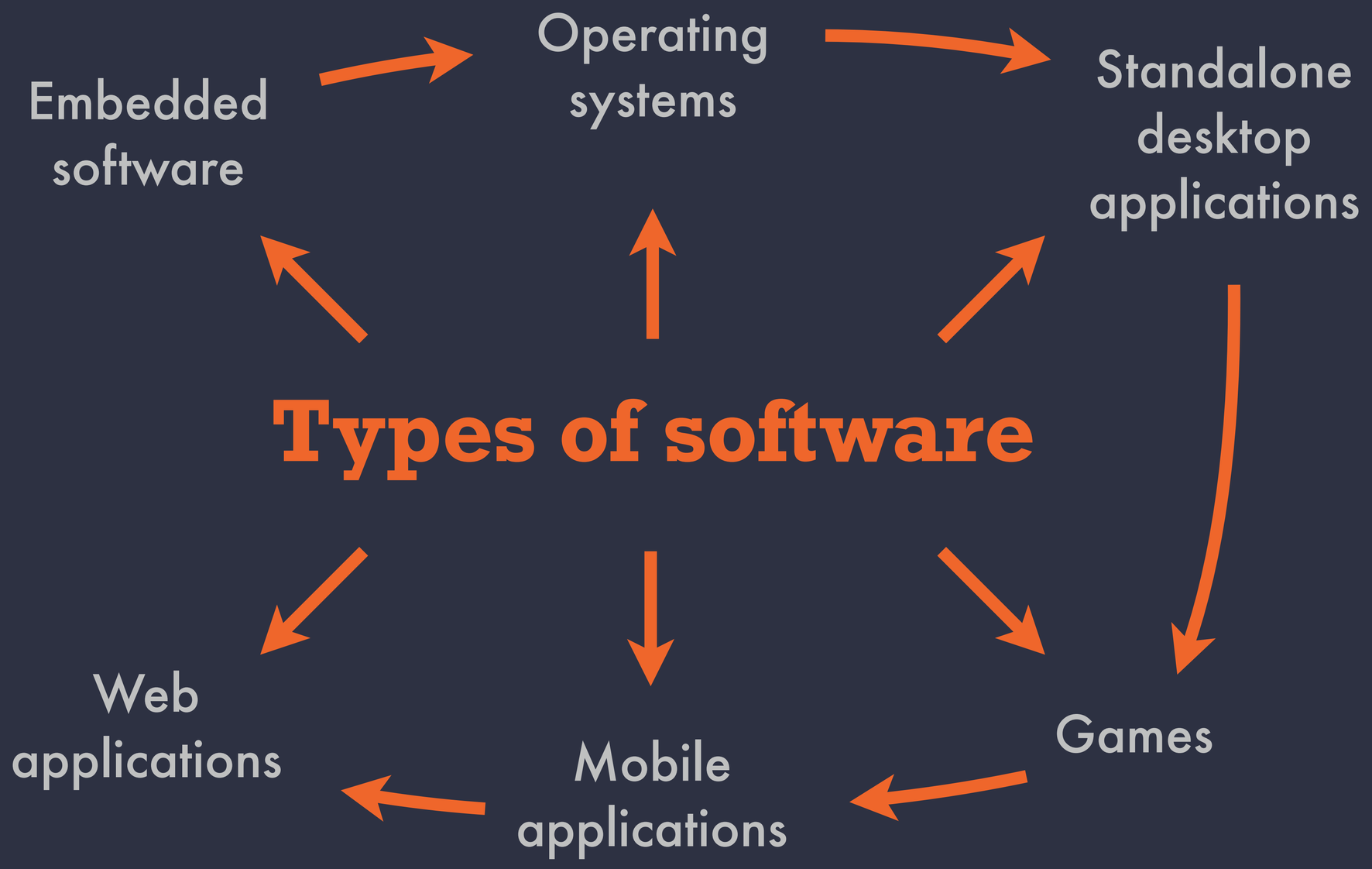
Standalone desktop applications

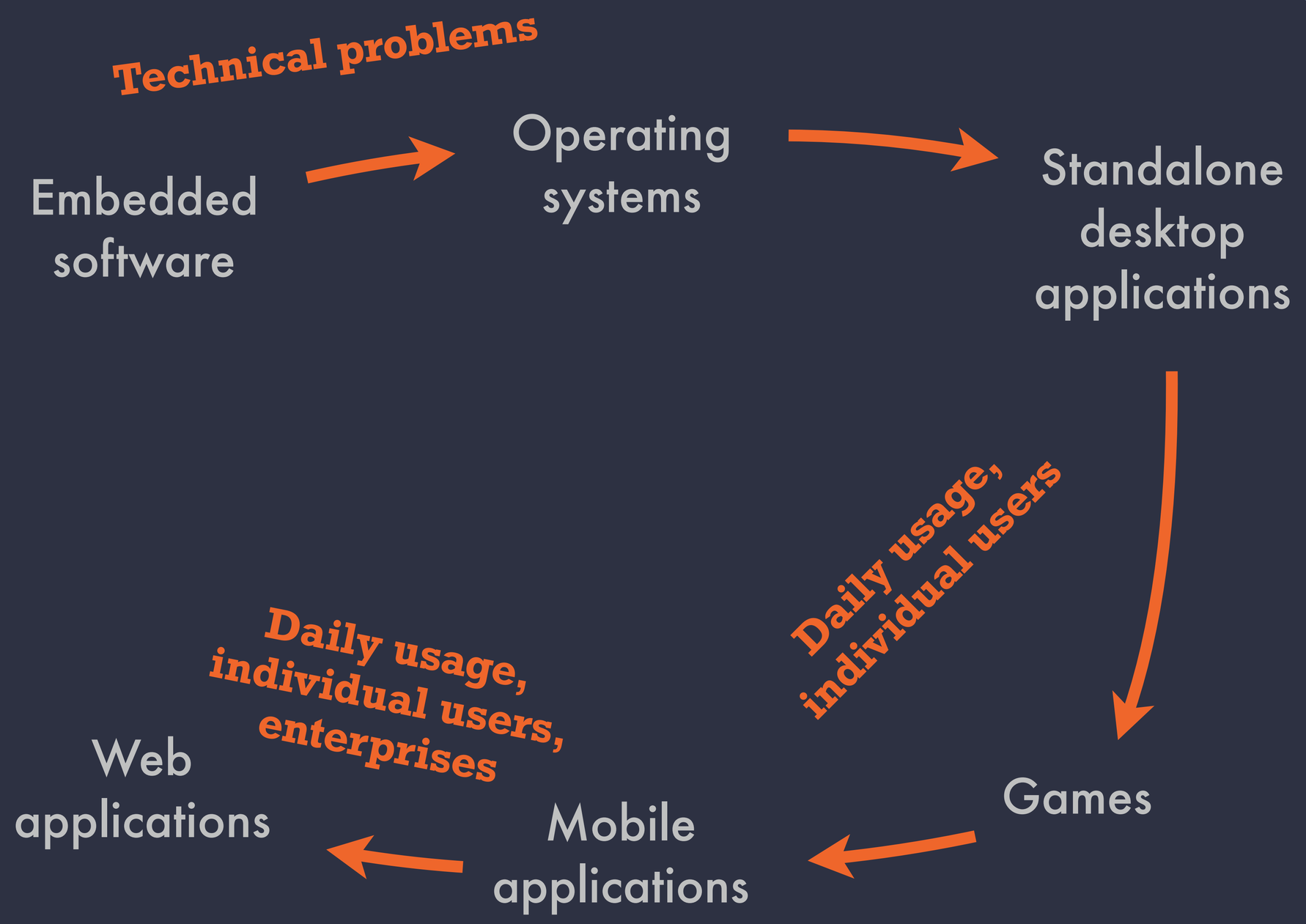
Web applications

Mobile applications

Games

Types of software




Technical problems
Embedded software
Operating systems
Standalone desktop applications
Daily usage, individual users
Games
Mobile applications
Daily usage, individual users, enterprises
Web applications

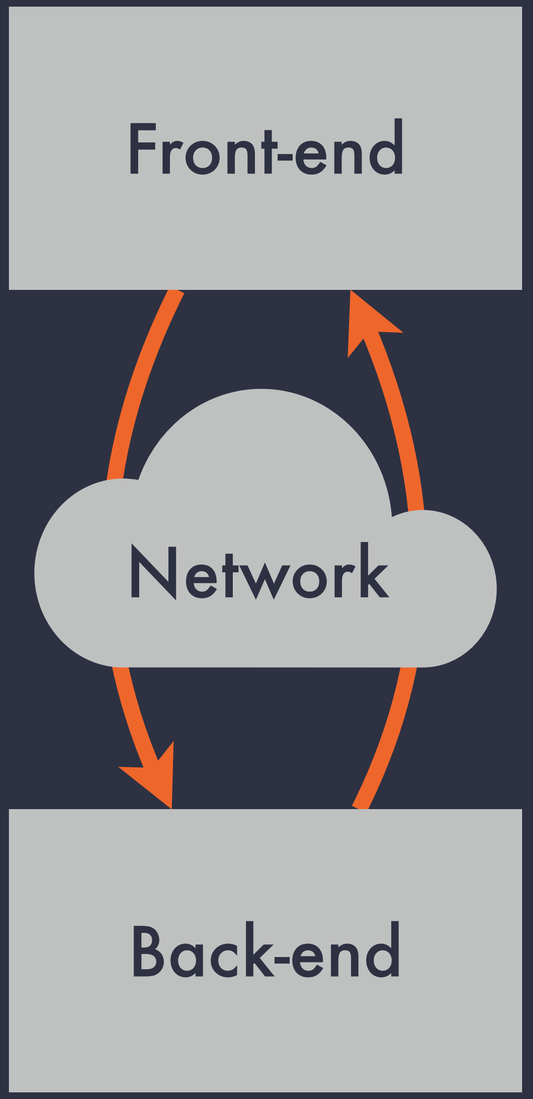
Web Applications

for Enterprises

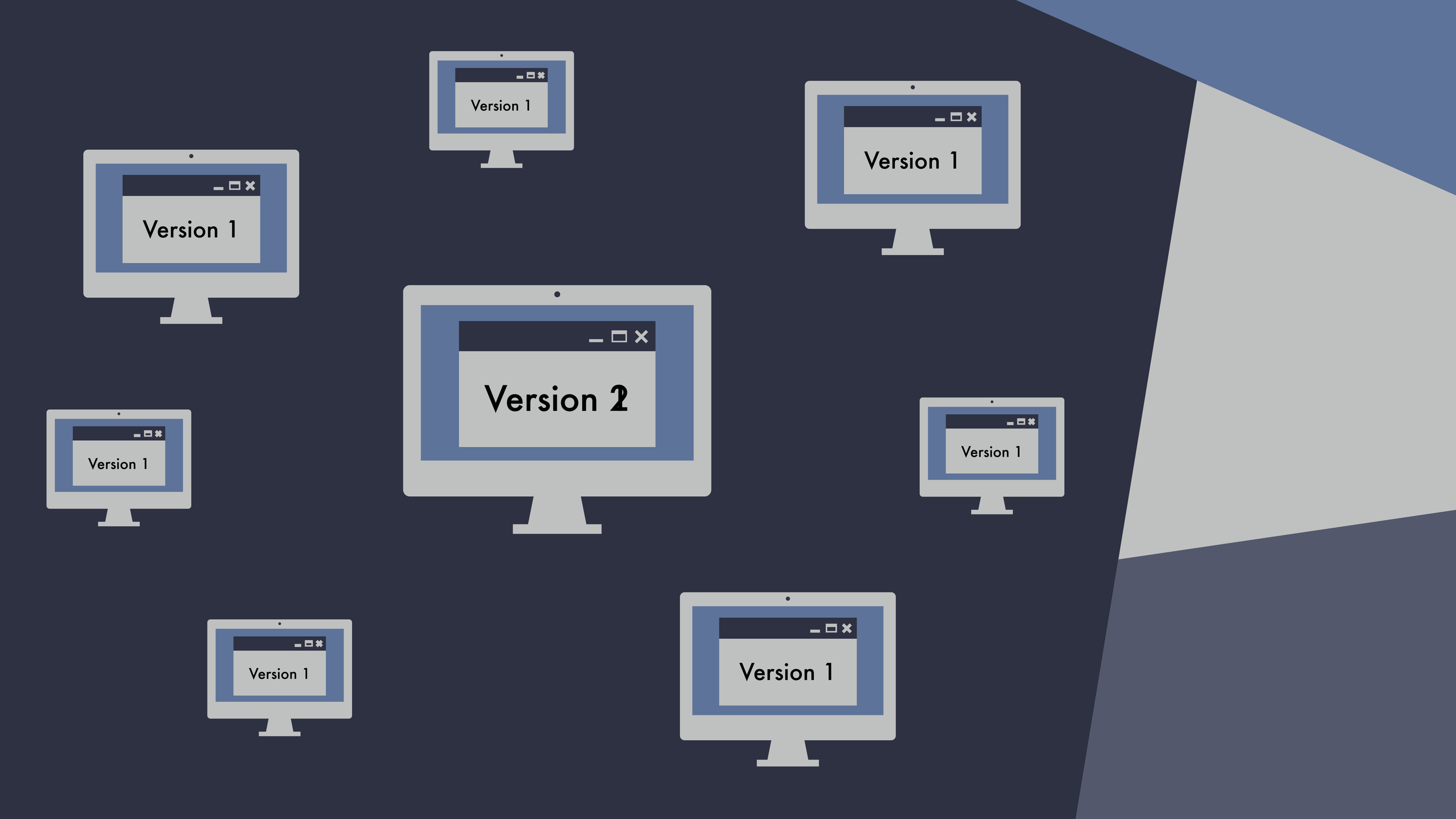
Enterprise software (...) is used to satisfy the needs of an organisation rather than individual users.

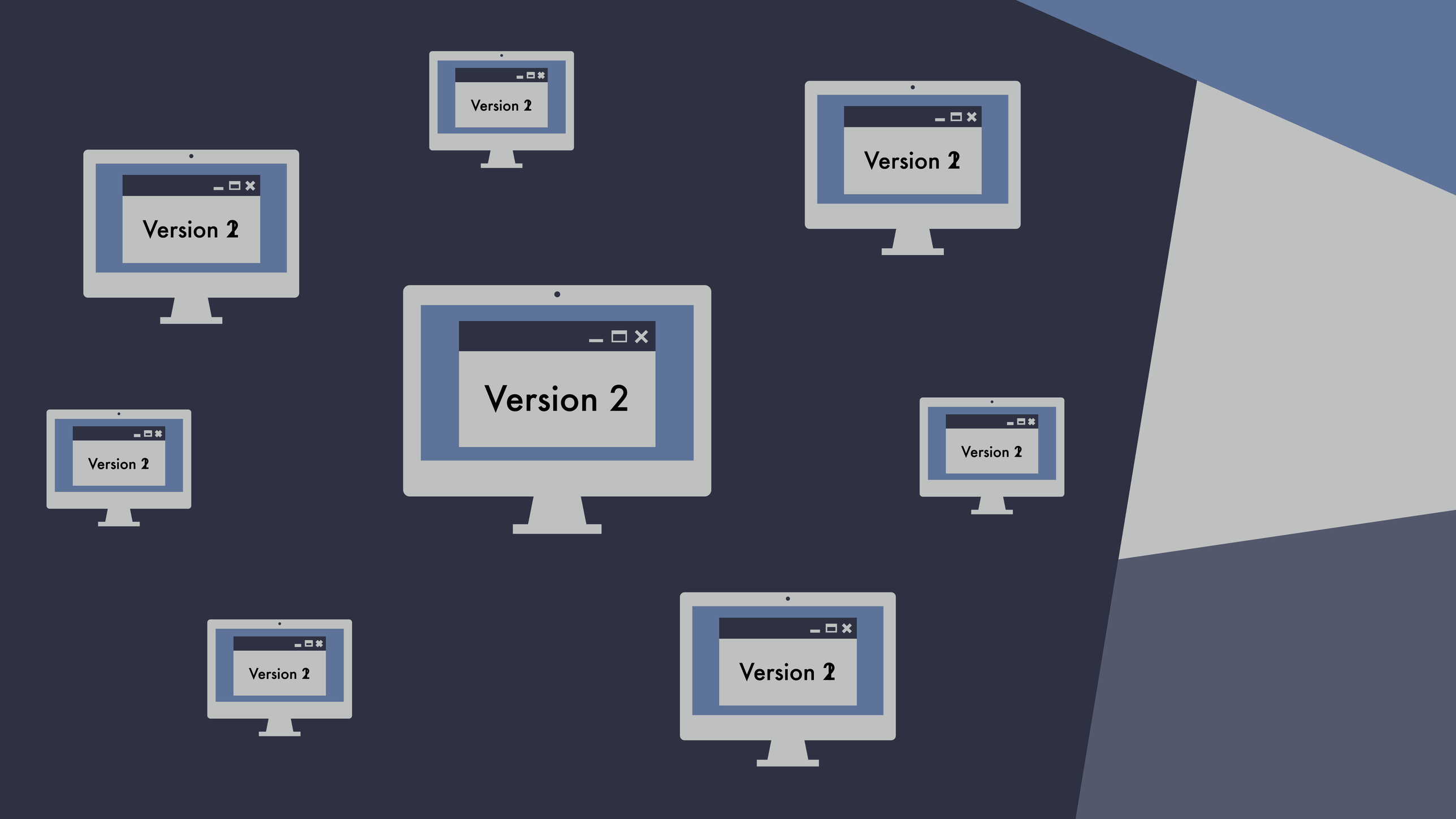
https://en.wikipedia.org/wiki/Enterprise_software

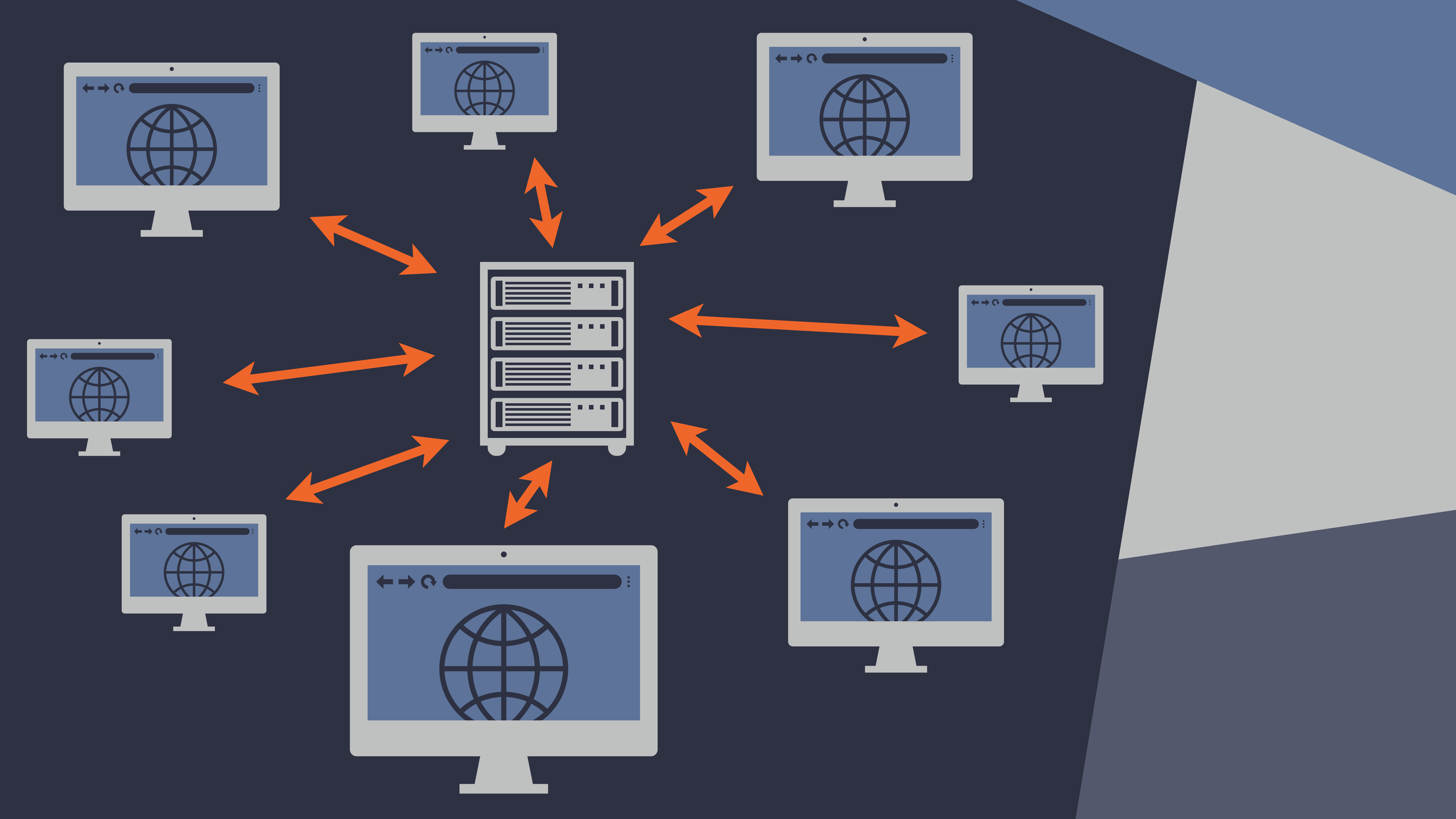
Web applications



Version 1


Version 1
Version 1
Version 1
Version 1
Version 2
Version 1
Version 1
Version 1


Version 2
Version 2
Version 2
Version 2
Version 2
Version 2
Version 2
Version 2
Version 2

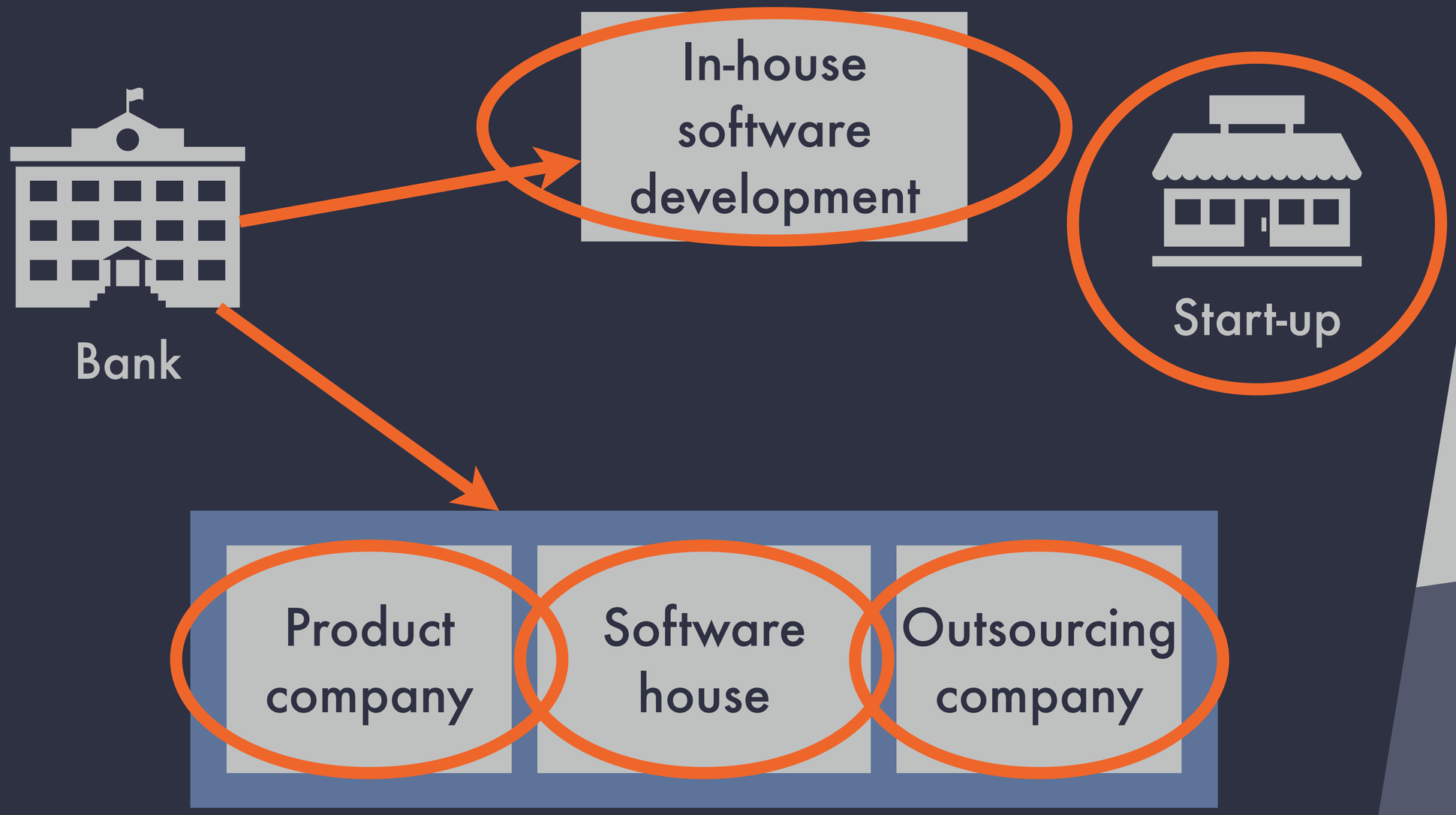


Desktop applications → **Web applications**

Long, long time ago → Now

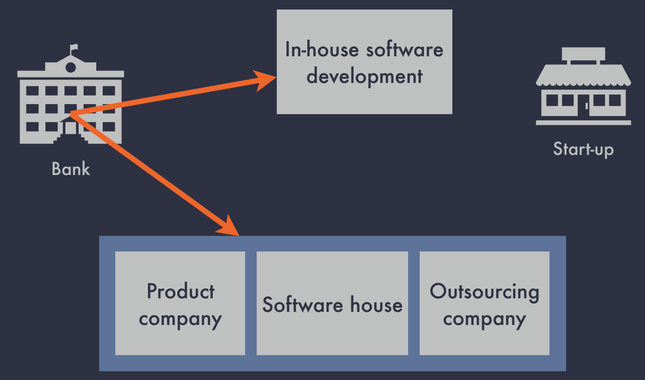
Types of Companies

That Develop Software


Bank
In-house software development
Start-up
Product company
Software house
Outsourcing company

Maturity

Possibility to learn best practices



Size

Willingness to invest in trainings

Requirements for a particular technology

Product focus

Long-term vs more varied experience

Being part of a big project

Your expertise and non-technical interests

Software Development

How Does it Work?

Business

↓

Requirements

↓

Software Development

↓

Running software

Insurance company

Software vendor

This is what we need.

It will reduce your costs and will cost you $40 mln.

GUARANTEED!

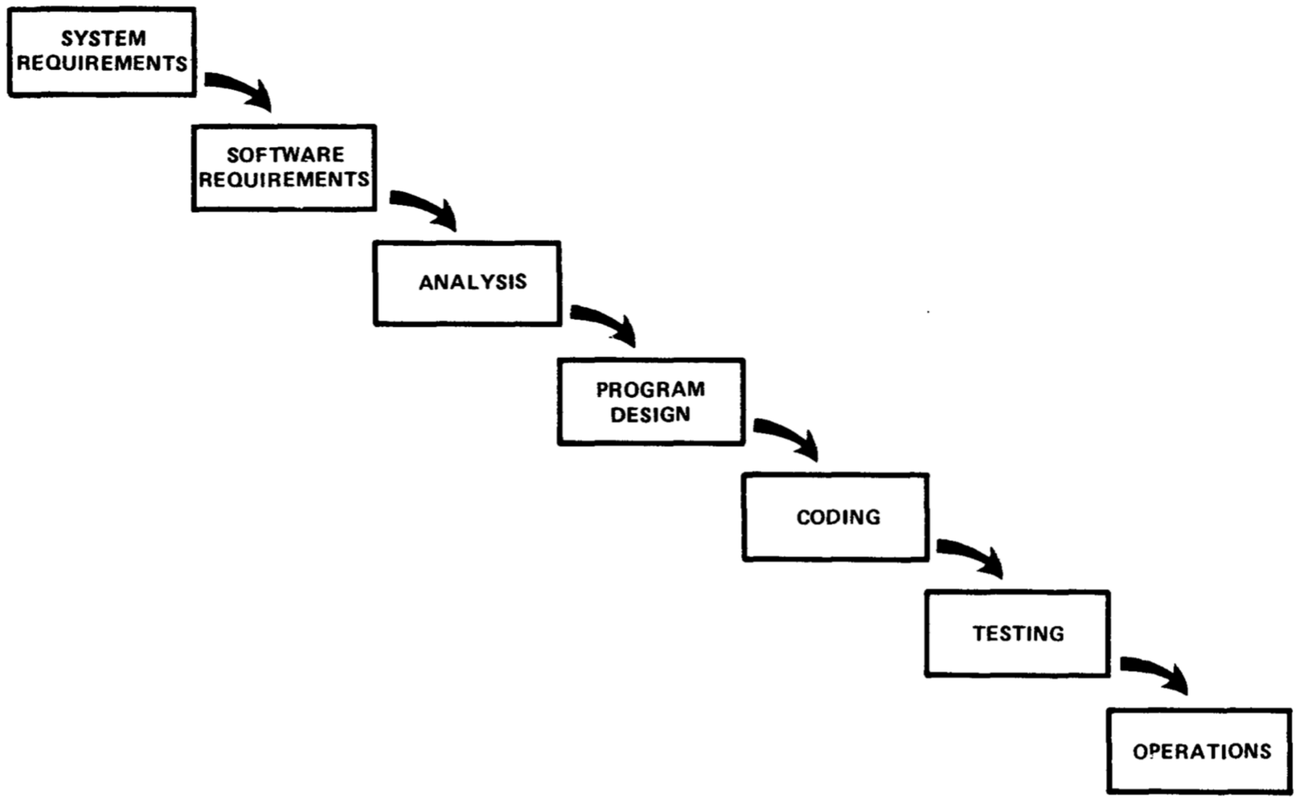


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

dr. Winston W. Royce, *Managing the Development of Large Software Systems*, 1970

I believe in this concept, but the implementation described above is risky and invites failure.

Managing the Development of Large Software Systems
dr. Winston W. Royce, 1970

UML
Analysis document
Design document
Working software

Late projects

Useless features

Late projects **Useless features**

Waterfall

1970

Scrum
XP
and more

Mid 90’

Manifesto for Agile Software Development

February 2001

Scrum

needs

XP (Extreme Programming)

practices to work.

Module 2

Daily Work

First Days

Onboarding training

Bootcamp

Computer unboxing and setup

Kitchen and coffee machine

Development environment setup

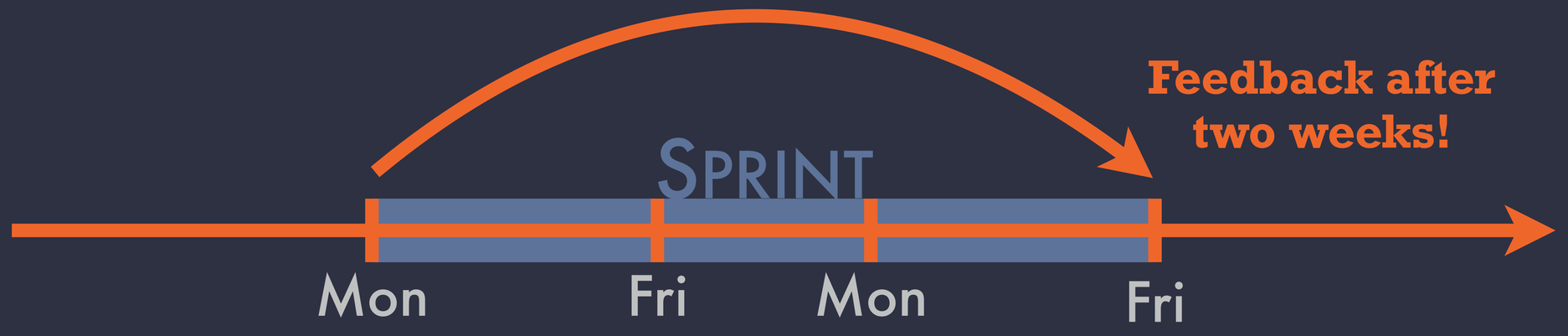
Project welcome

Scrum

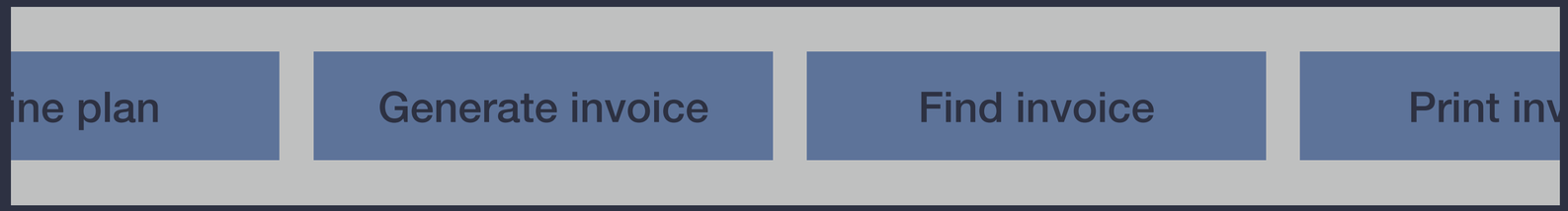
Scrum is lightweight, simple to understand, difficult to master.

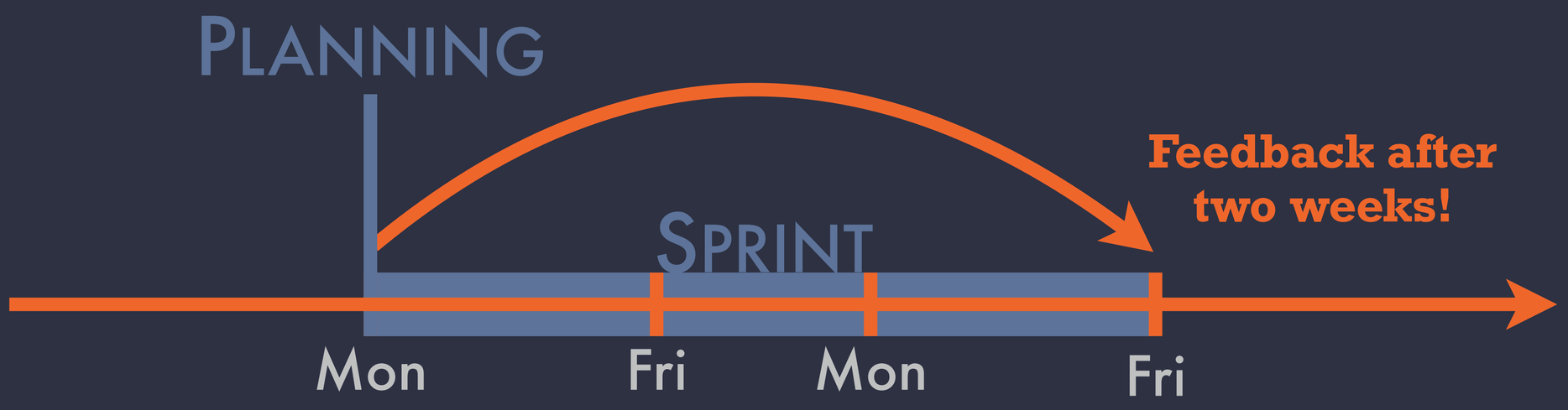
The Scrum Guide
Ken Schwaber and Jeff Sutherland

Feedback only at the very end!
Now
2 years later


Feedback after two weeks!
SPRINT
Mon
Fri
Mon
Fri

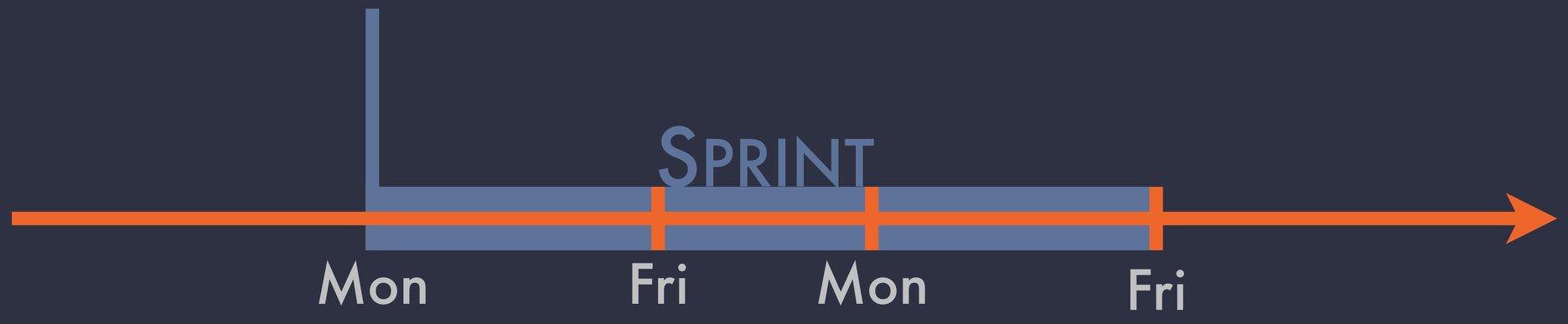
Backlog

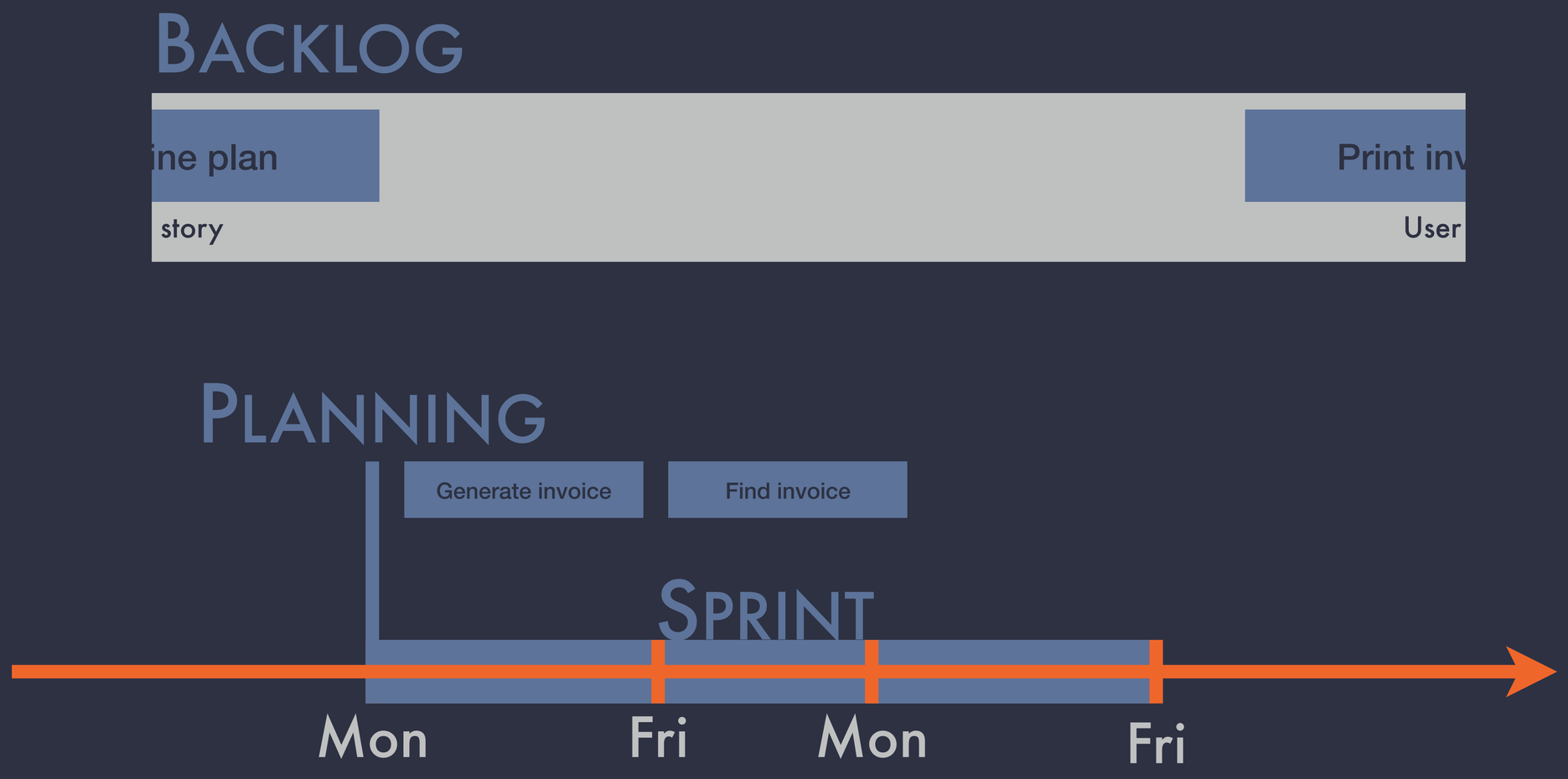

ine plan
Generate invoice
Find invoice
Print inv

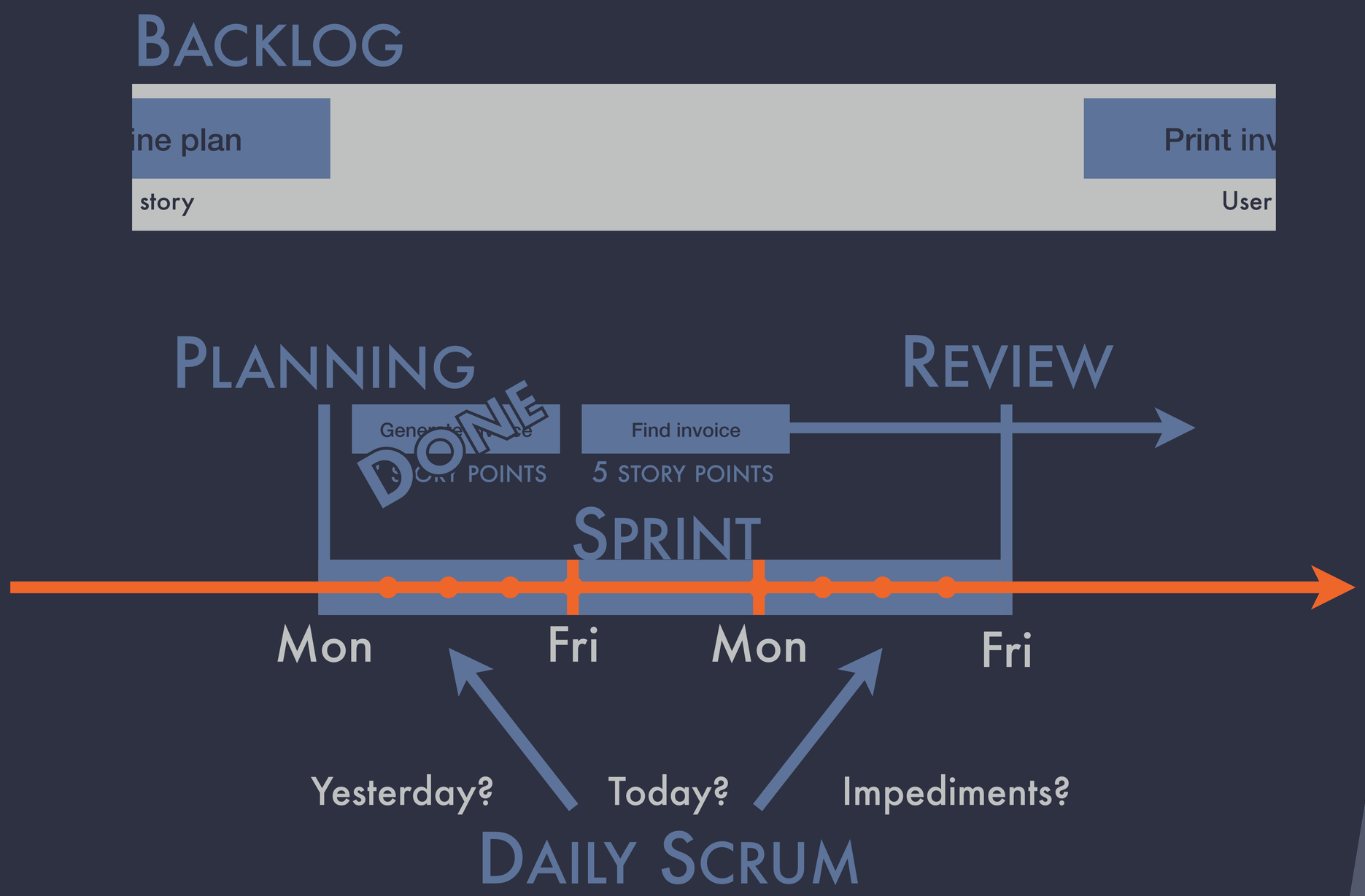

Planning
Sprint
Feedback after two weeks!
Mon
Fri
Mon
Fri

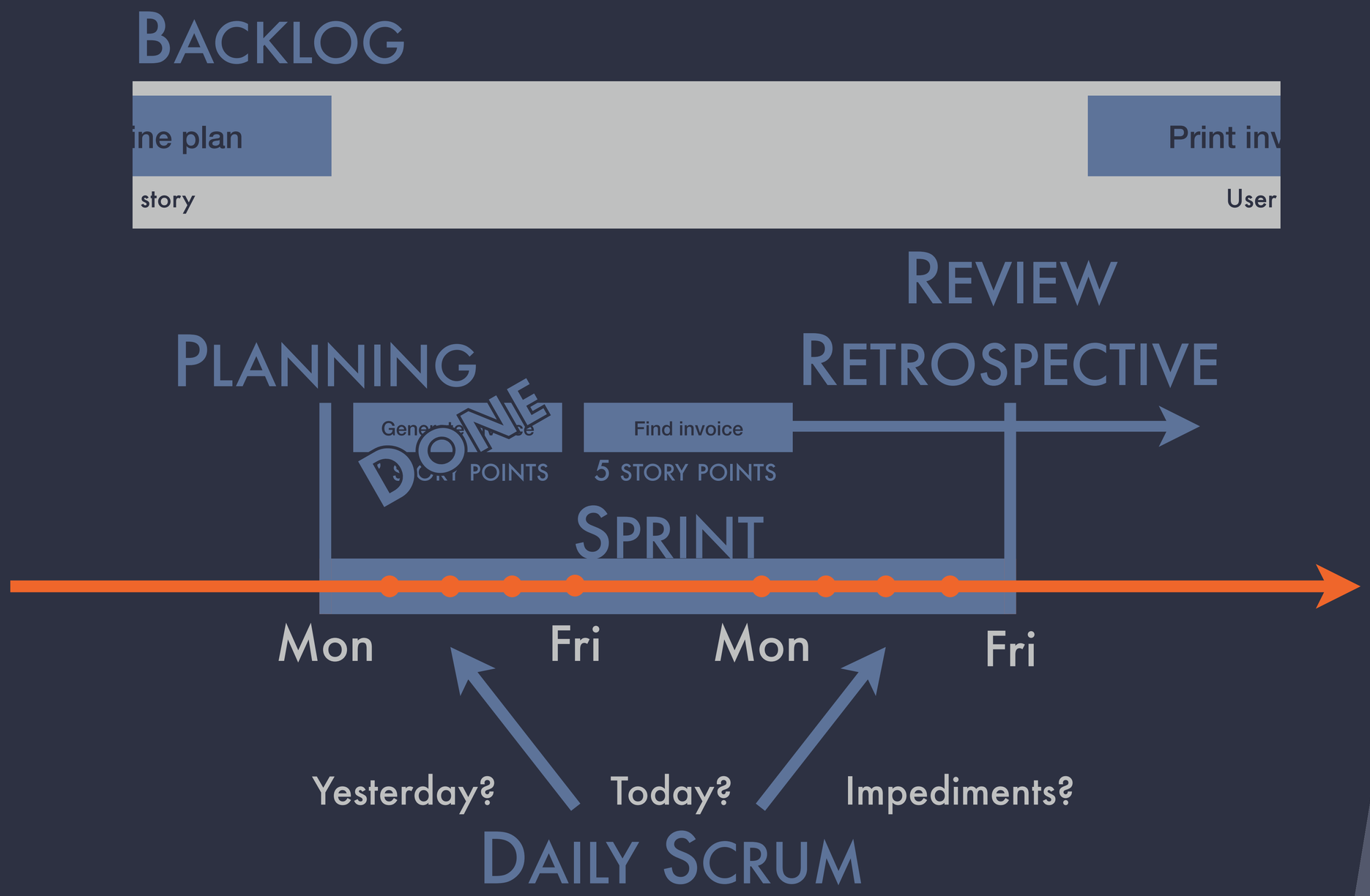
Backlog

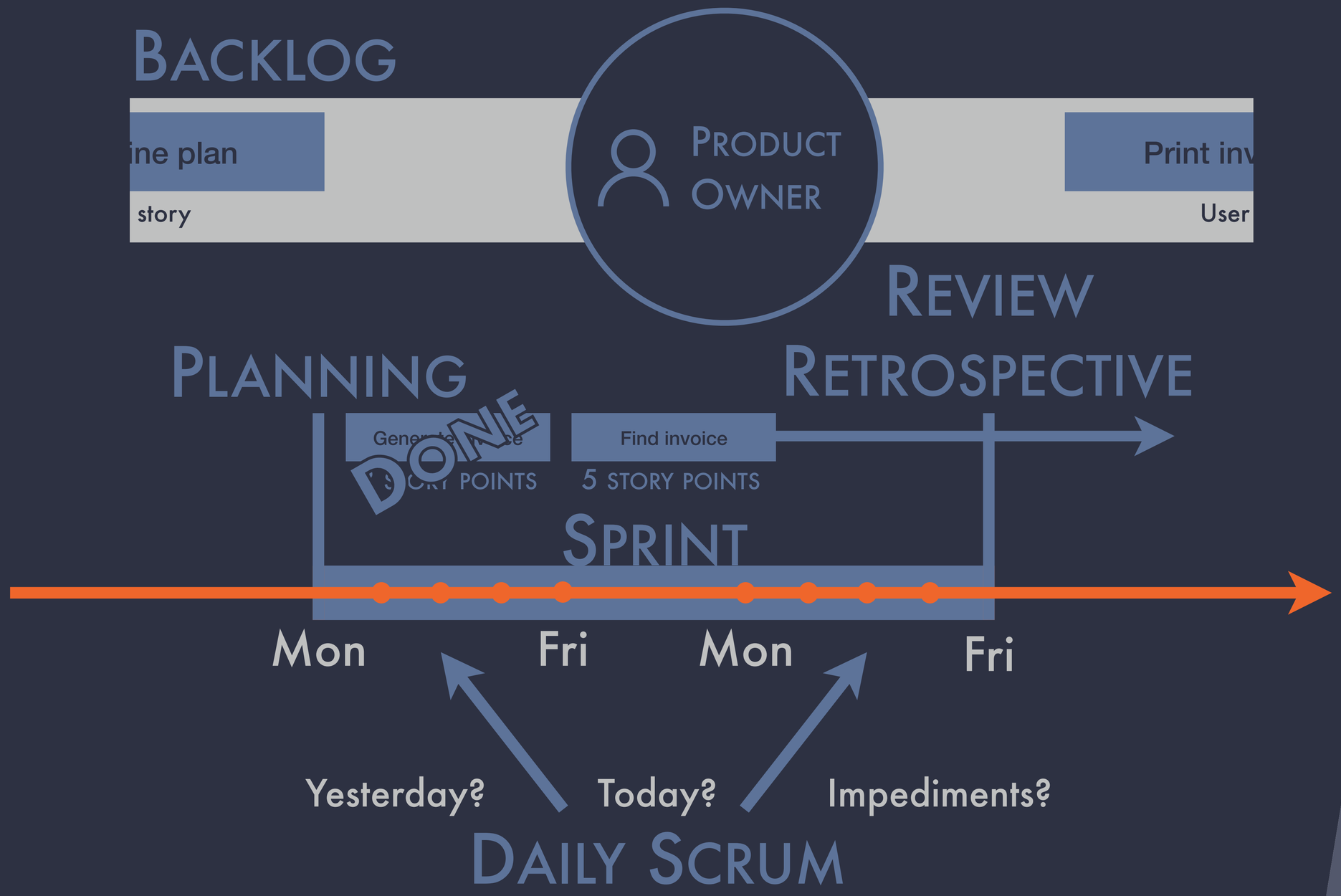
Planning




Backlog
ine plan
Print inv
story
User
Planning
Generate invoice
Find invoice
Sprint
Mon
Fri
Mon
Fri


Backlog
ine plan
Print inv
story
User
Planning
Review
Find invoice
DONE
5 STORY POINTS
Sprint
Mon
Fri
Mon
Fri
Yesterday?
Today?
Impediments?
Daily Scrum


Backlog
ine plan
Print inv
story
User
Review
Planning
Retrospective
Find invoice
DONE
5 STORY POINTS
Sprint
Mon
Fri
Mon
Fri
Yesterday?
Today?
Impediments?
Daily Scrum


Backlog
ine plan
story
Product Owner
Print inv
User
Review
Planning
Retrospective
Generate invoice
DONE
Find invoice
5 story points
Sprint
Mon
Fri
Mon
Fri
Yesterday?
Today?
Impediments?
Daily Scrum

Backlog

ine plan

story

Product Owner

Print inv

User

Devs
DBAs
UXs
QAs
BAs

Scrum Master

Sprint

3 - 9 team members
+ Scrum Master
+ Product Owner

User Story

Backlog

ine plan	Generate invoice	Find invoice	Print inv
story	User story	User story	User

User Story

As a <*who? a type of a user*>,
I want <*what?: some action to happen*>,
so that <*why?: a benefit or a value*>.

Independent
Negotiable
Valuable
Estimable
Small
Testable

Scrum

needs

XP (Extreme Programming)

to work.

Scrum

Sprint

properly defined user stories

INVEST

XP (Extreme Programming)

PREPARATION

EXECUTION

Scrum

needs both

INVEST and XP (Extreme Programming)

to work.

Story Points

Story Points

?

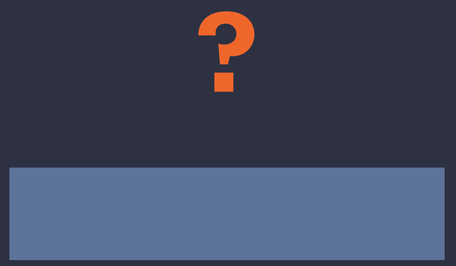
Find invoice

?

Generate invoice

Story Points

Find invoice



Generate invoice

Story Points

3 story points

Find invoice

5 story points

Generate invoice

Each sprint has a goal.

Stories completed in the sprint need to be done.

3 story points

Find invoice

5 story points

Generate invoice

Each sprint has a goal.

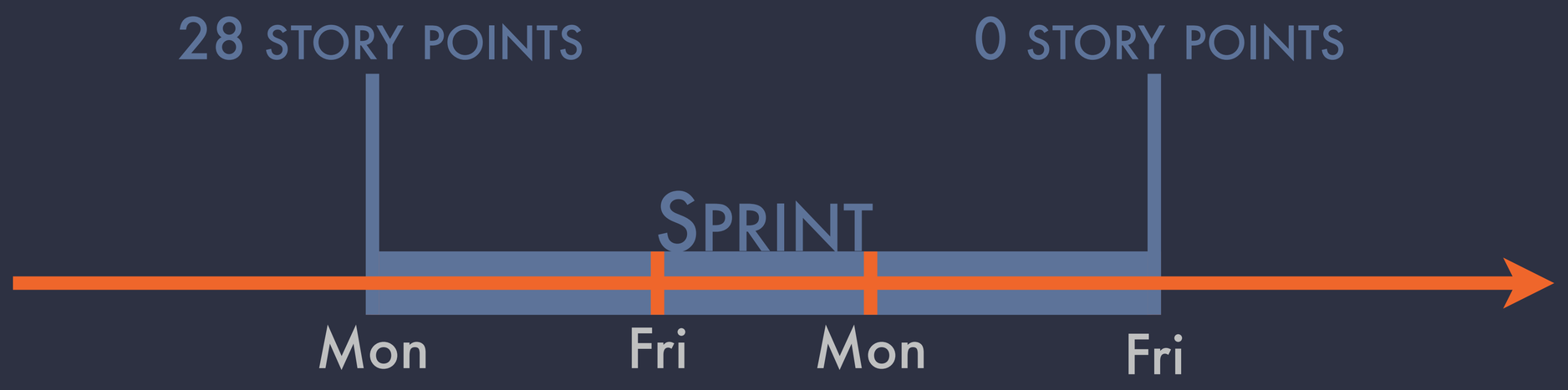
Stories completed in the sprint need to be <u>DONE</u>.

Definition of "Done"

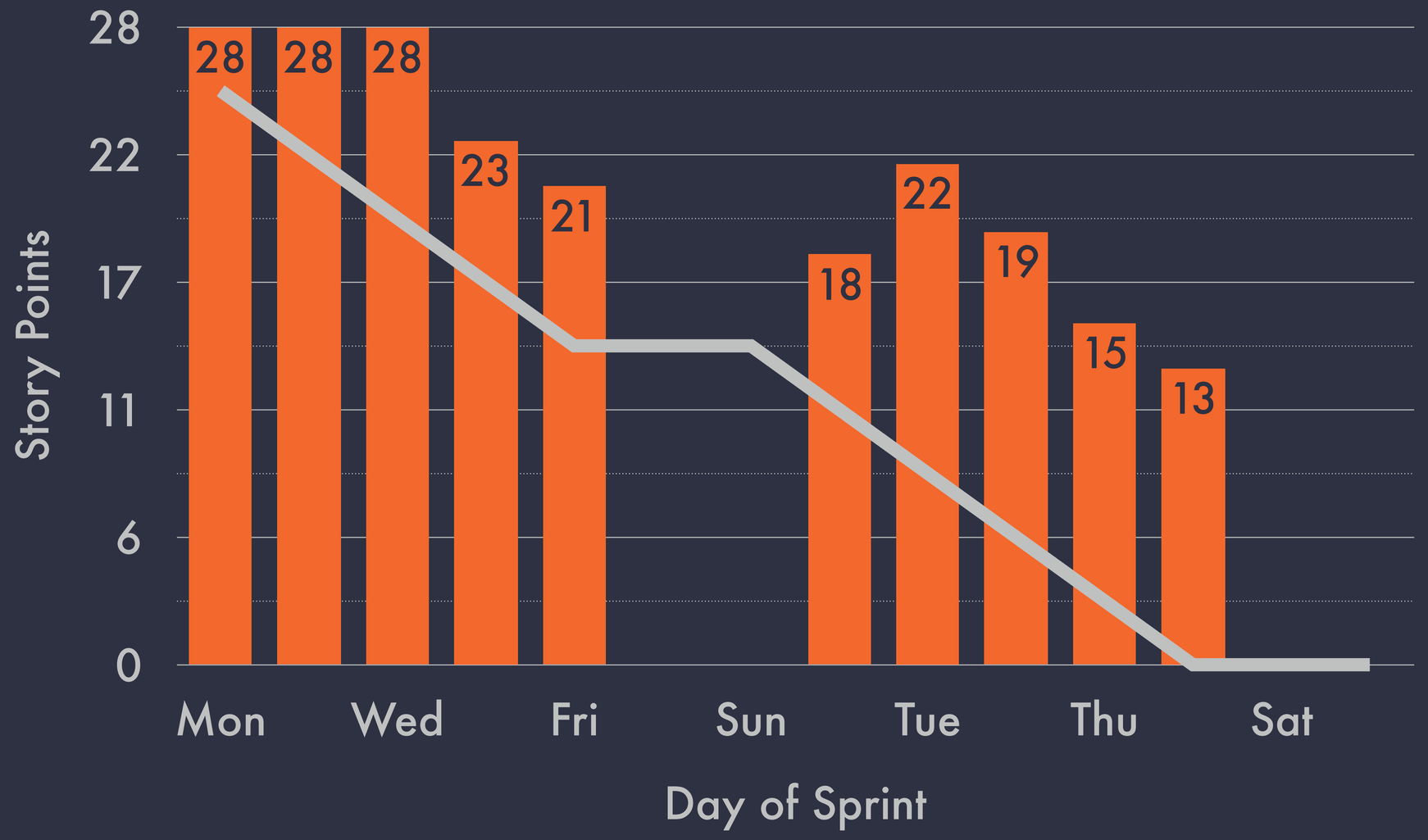
Find invoice

Generate invoice

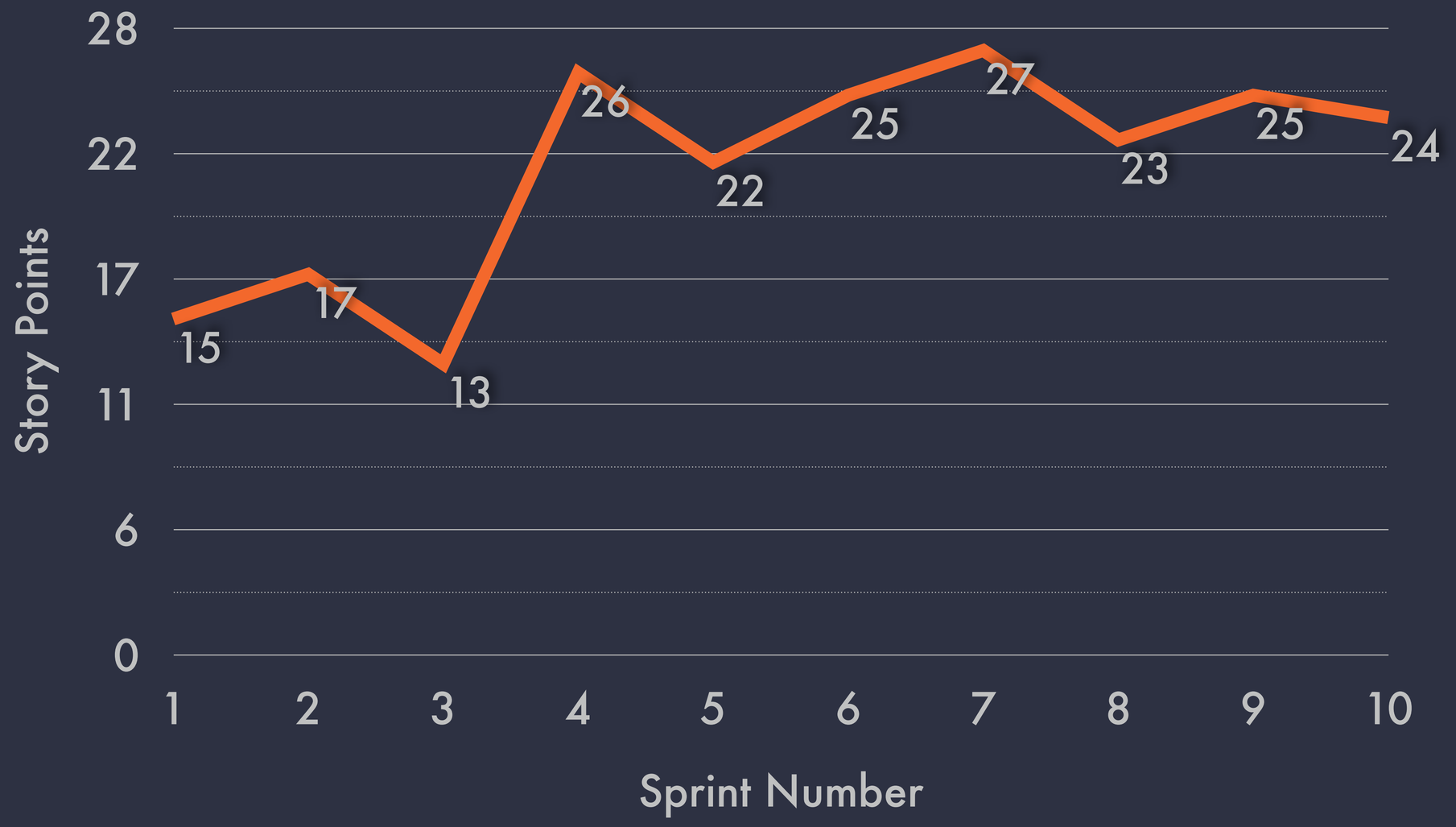
One more point regarding Scrum…


28 STORY POINTS
0 STORY POINTS
SPRINT
Mon
Fri
Mon
Fri

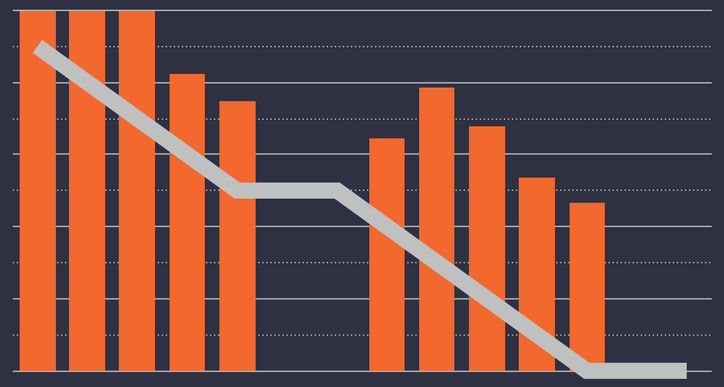
Burndown


28
22
17
11
6
0
Story Points
28
28
28
23
21
18
22
19
15
13
Mon
Wed
Fri
Sun
Tue
Thu
Sat
Day of Sprint

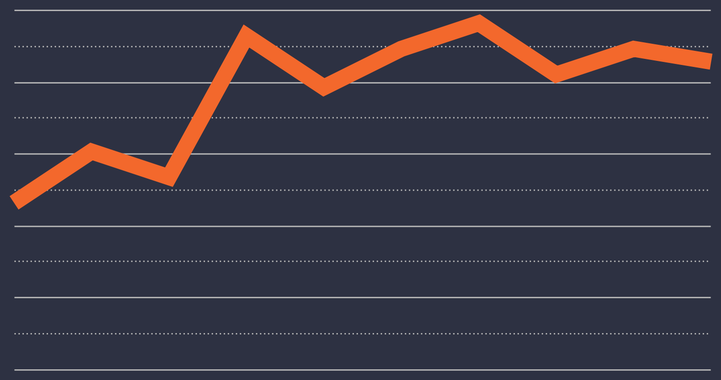
Velocity


28
22
17
11
6
0
Story Points
15
17
13
26
22
25
27
23
25
24
1
2
3
4
5
6
7
8
9
10
Sprint Number

Burndown



Velocity



Scrum is lightweight, simple to understand, difficult to master.

The Scrum Guide
Ken Schwaber and Jeff Sutherland

Tools

To do	In progress	Done
Find Invoice 3		
Generate Invoice 5		

crum board

Scrum board

To do	In progress	Done
Generate Invoice 5	Find Invoice 3 Alice	

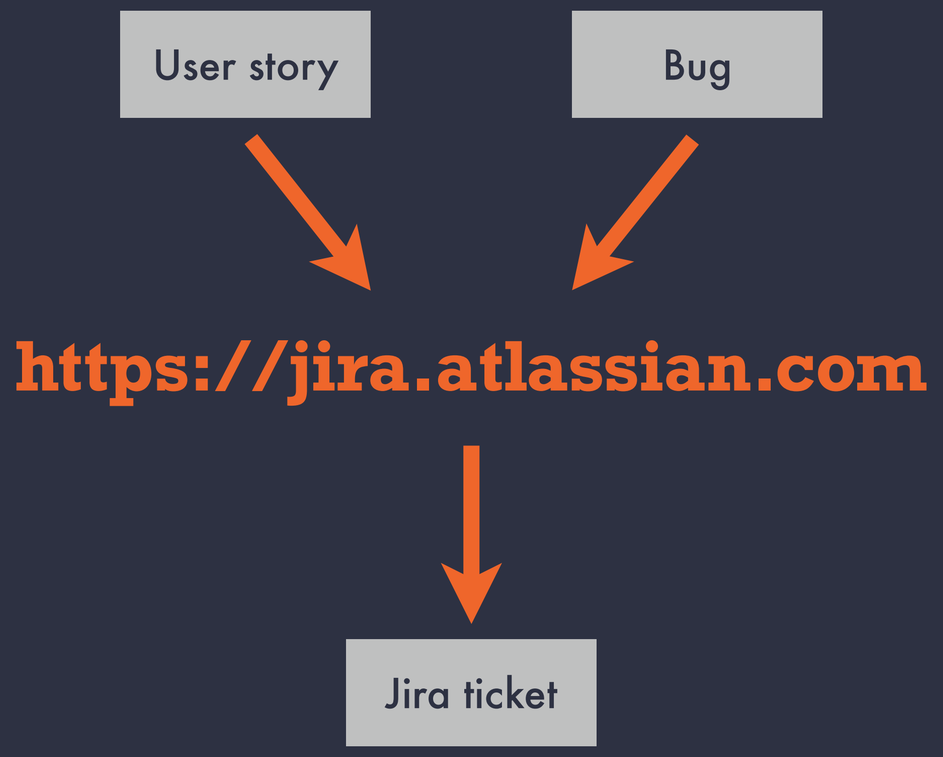
Scrum board

To do	In progress	Done
Generate Invoice 5		Find Invoice 3 Alice

https://trello.com

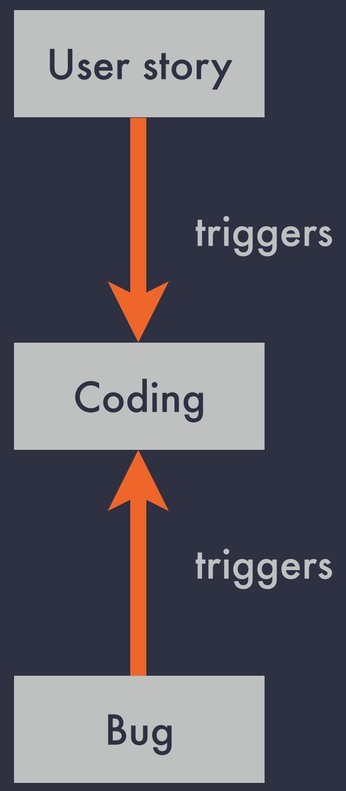
Scrum board

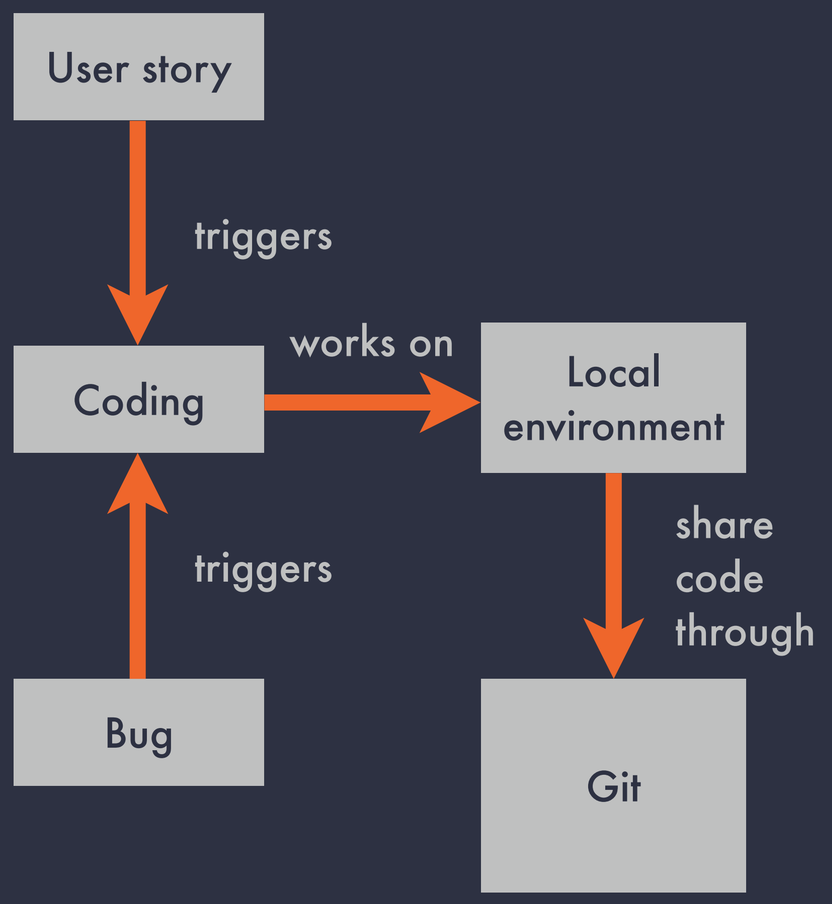
To do	In progress	In code review	Tested	Done
+ Backlog				
			+ Calculations	
	+ Charts			
+ Stories assignment				
		+ Planning support		

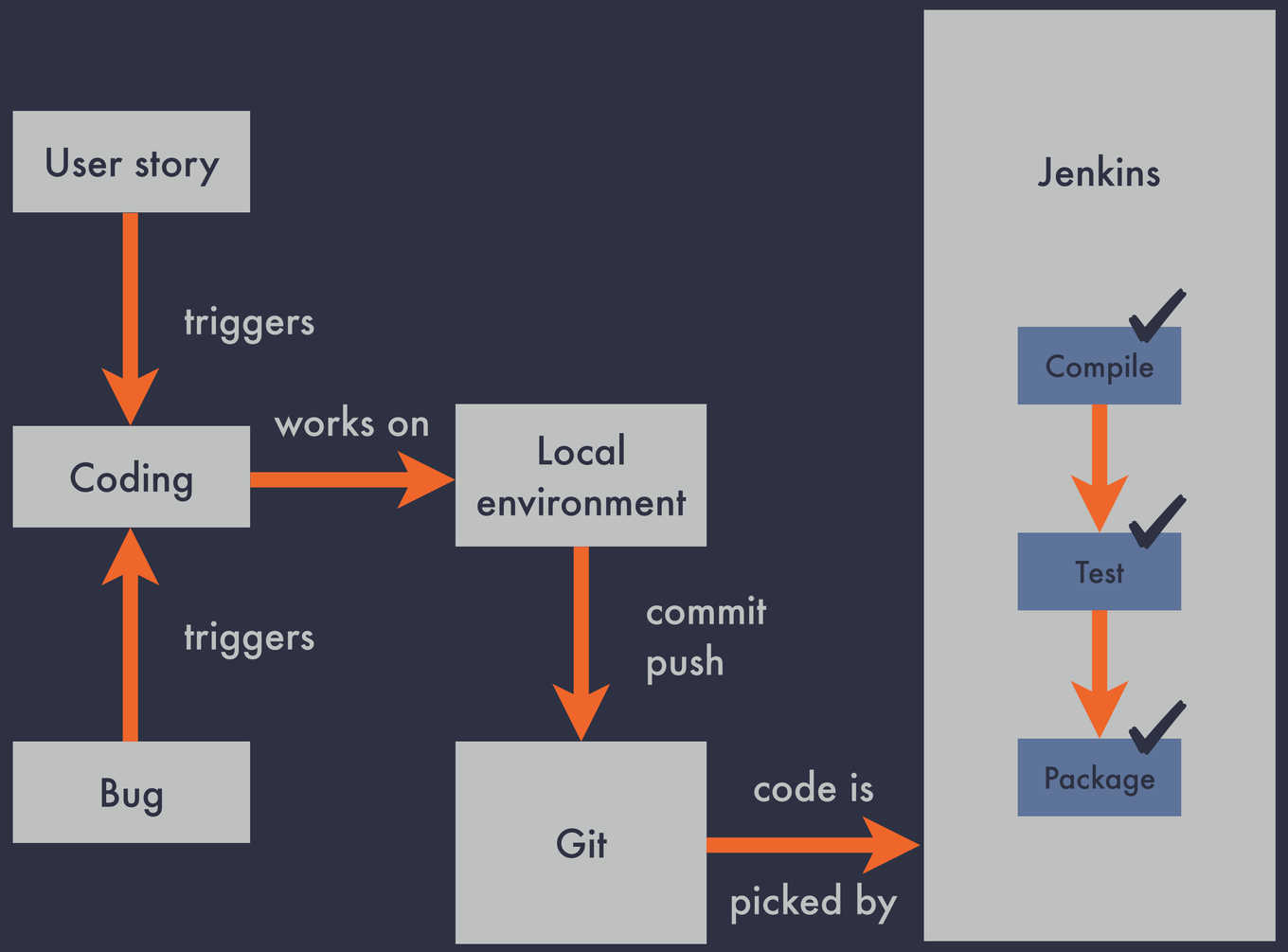

User story
Bug
https://jira.atlassian.com
Jira ticket

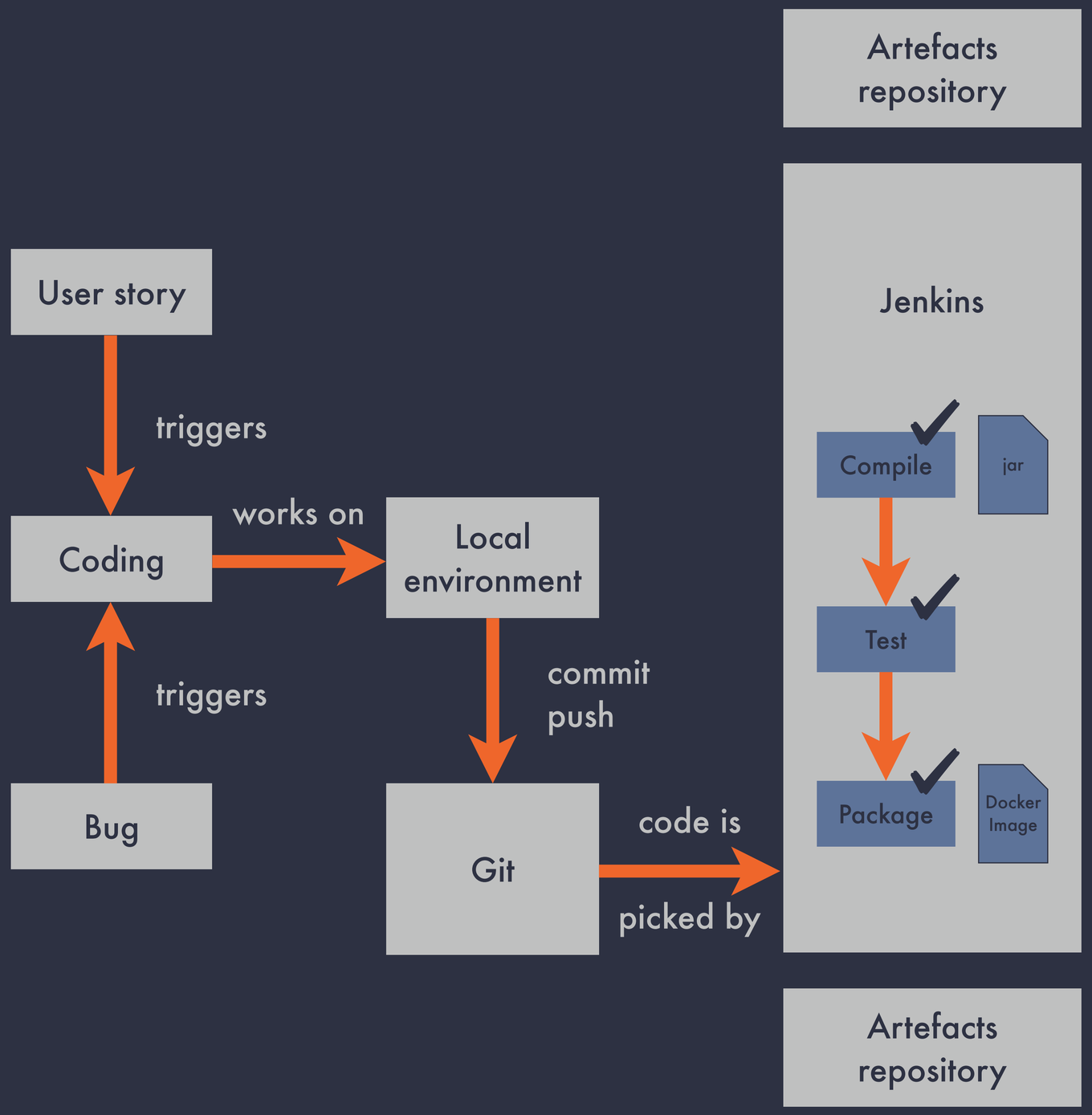
A Bug in Production!

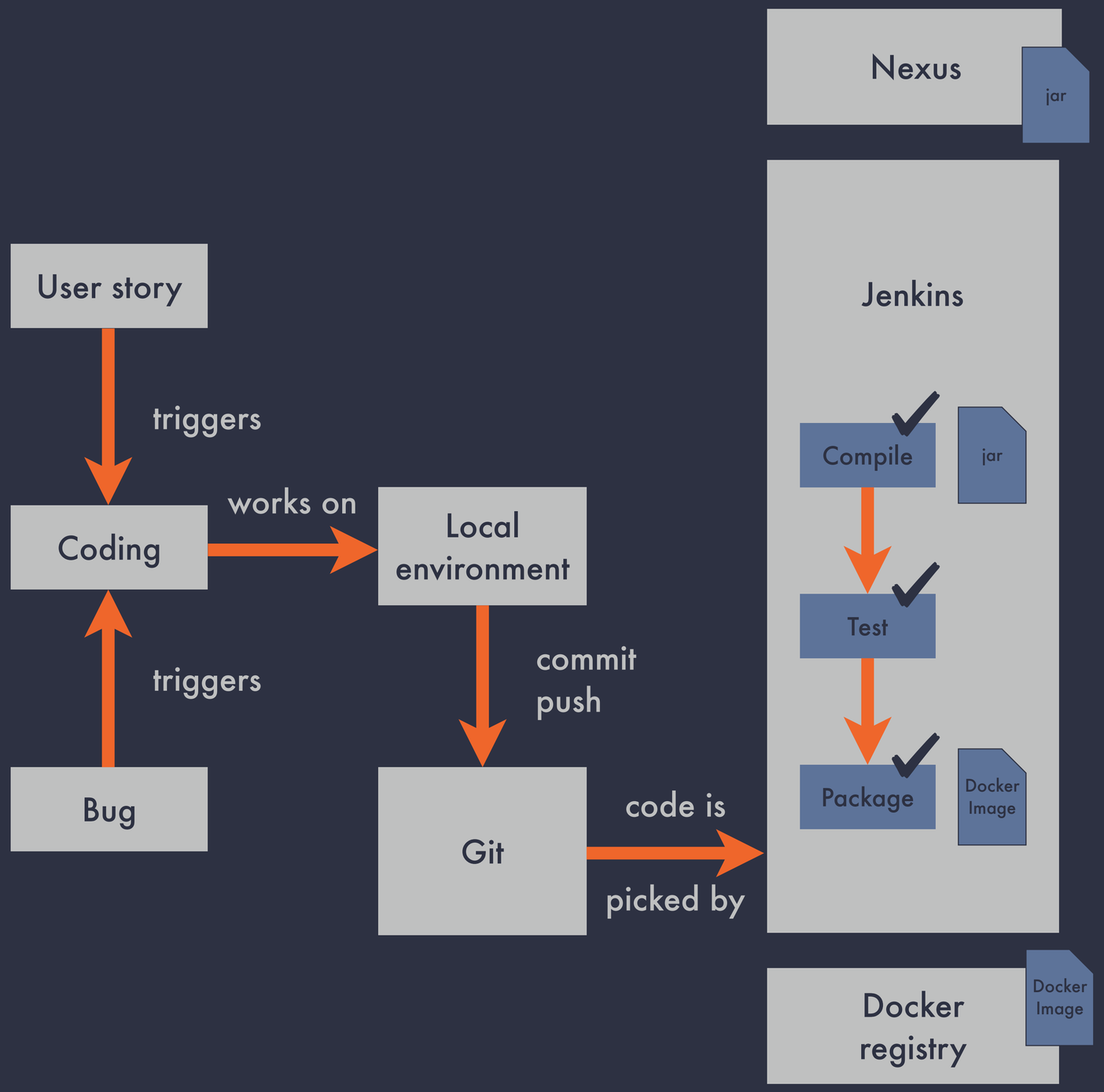
User story

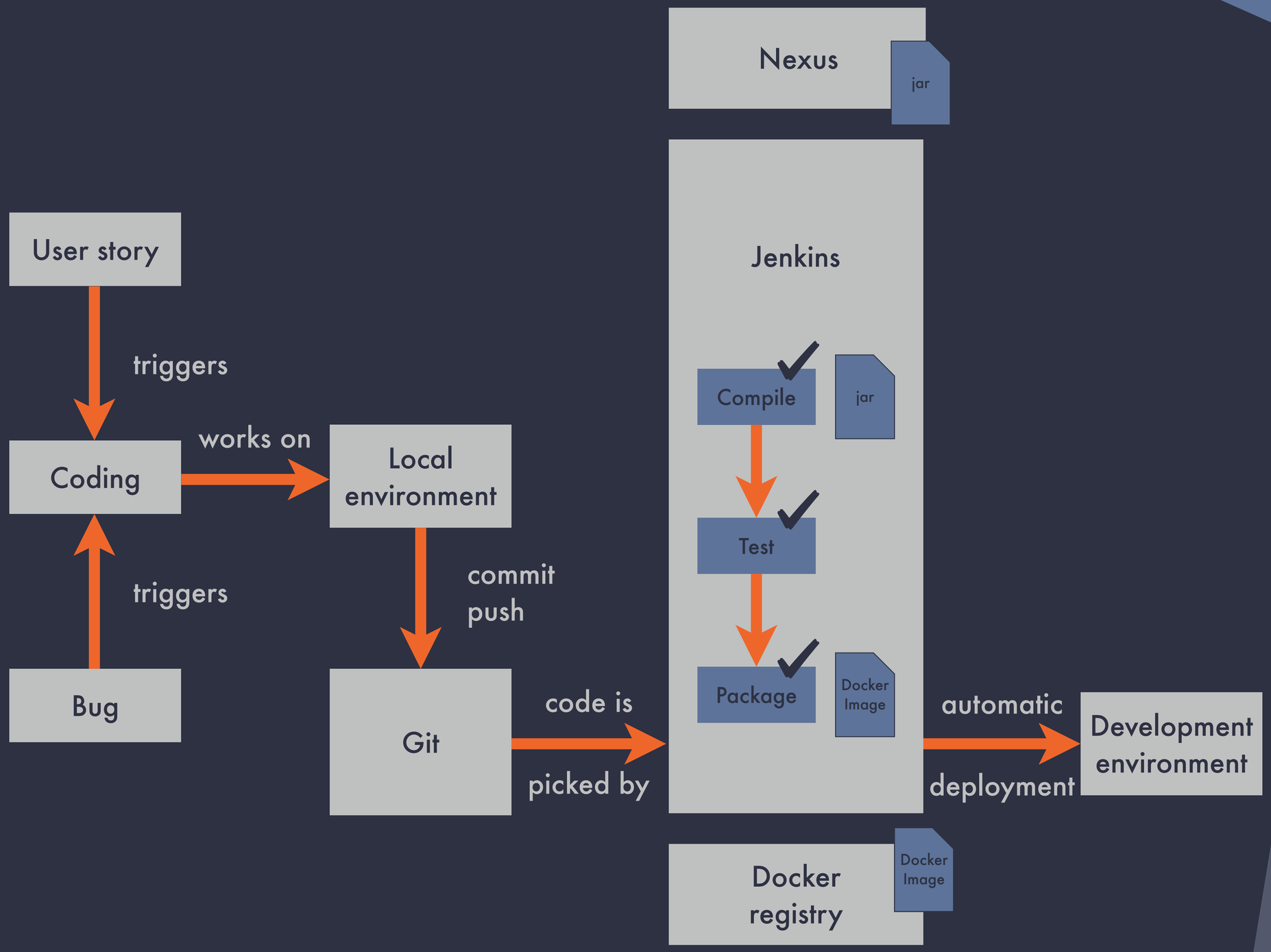

User story
triggers
Coding
triggers
Bug

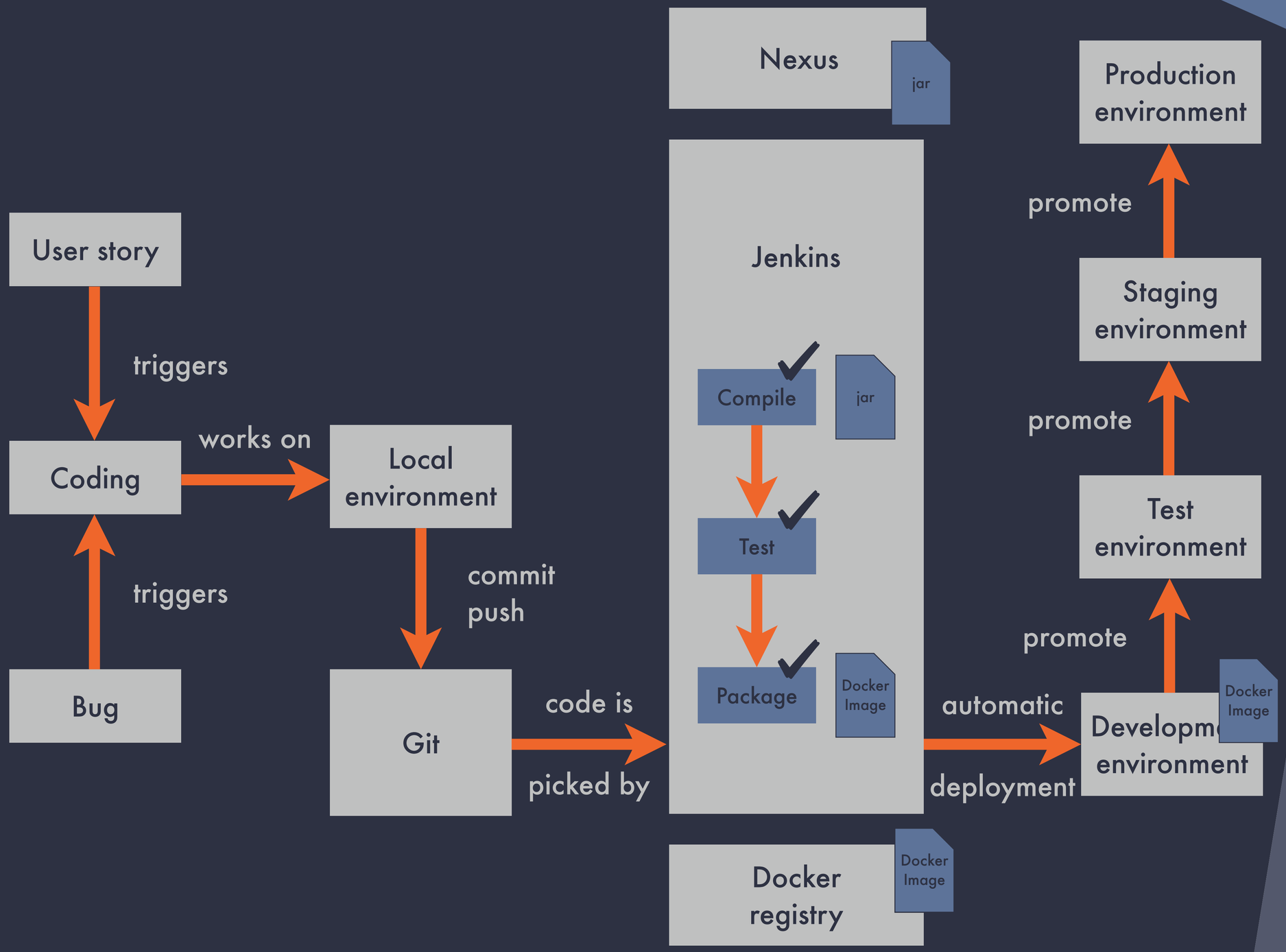

User story
triggers
Coding
works on
Local environment
share code through
Git
triggers
Bug

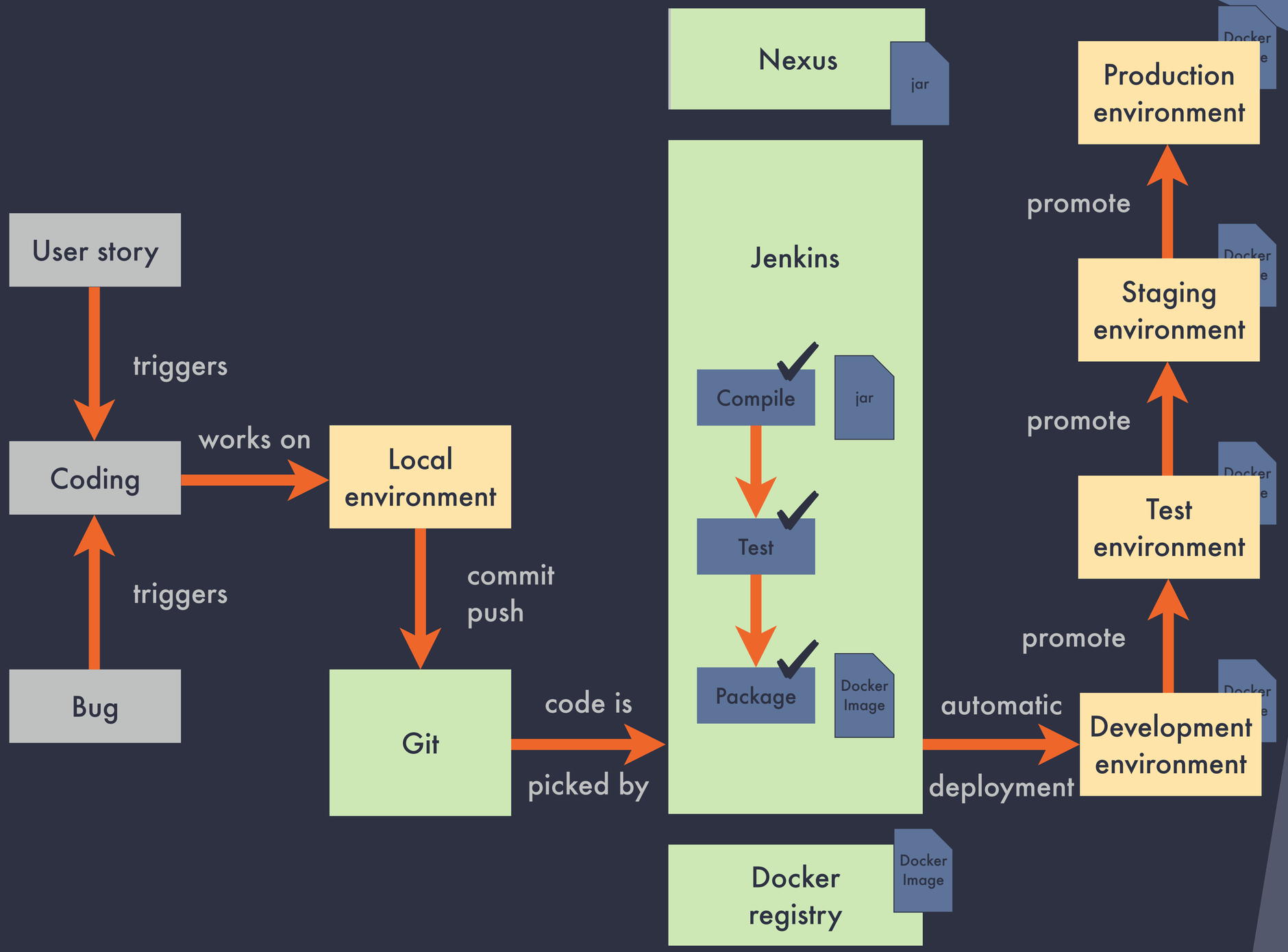

User story
triggers
Coding
works on
Local environment
commit
push
triggers
Bug
Git
code is
picked by
Jenkins
Compile
Test
Package

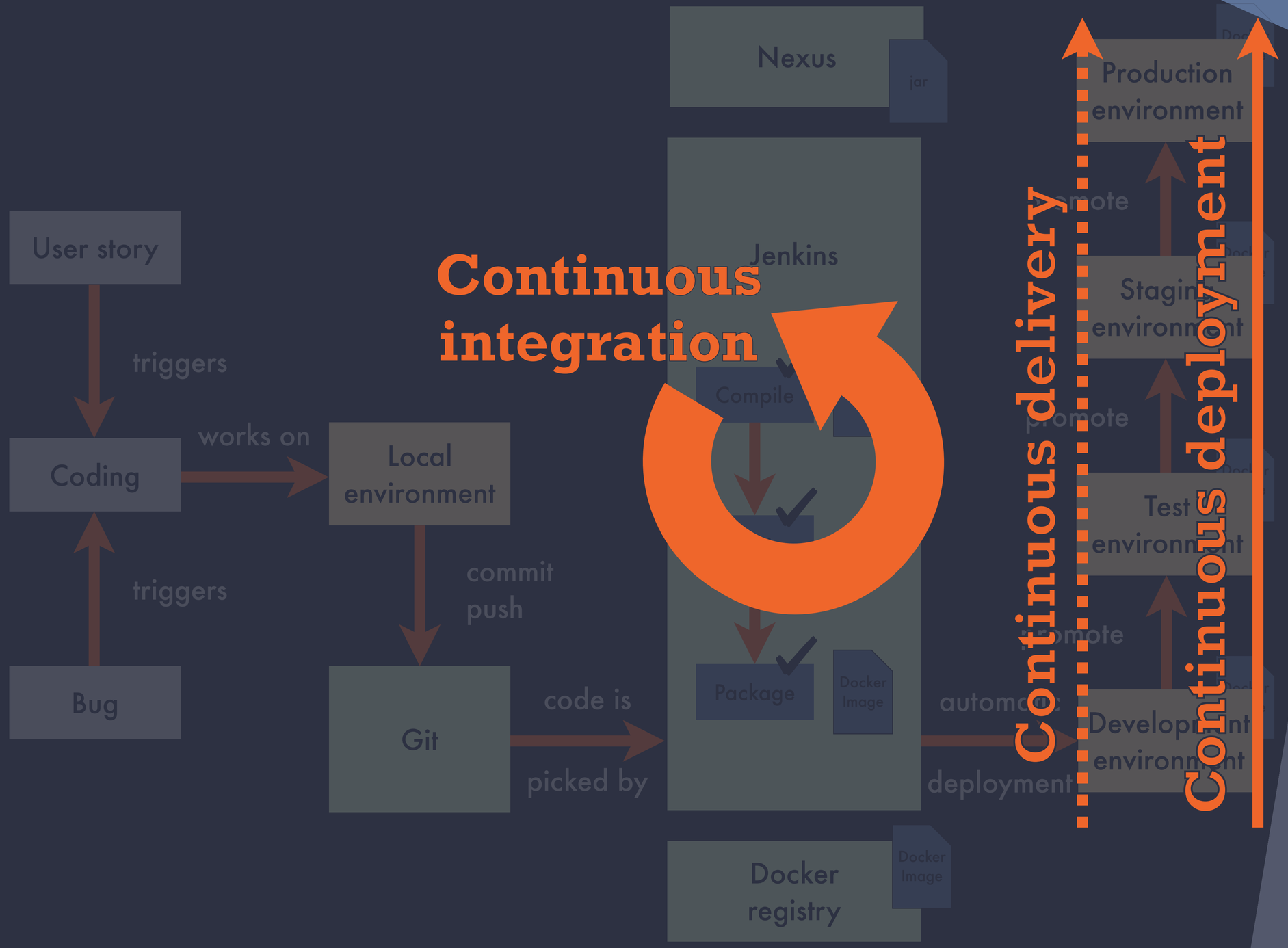

Artefacts repository
Jenkins
User story
triggers
Compile
jar
Coding
works on
Local environment
Test
commit
push
triggers
Bug
Package
Docker Image
Git
code is
picked by
Artefacts repository


Nexus
jar
User story
triggers
Jenkins
Compile
jar
Coding
works on
Local environment
Test
commit
push
triggers
Bug
Package
Docker Image
Git
code is
picked by
Docker registry
Docker Image


Nexus
jar
User story
triggers
Jenkins
Compile
jar
Coding
works on
Local environment
Test
commit
push
triggers
Bug
Package
Docker Image
Git
code is
picked by
automatic
deployment
Development environment
Docker registry
Docker Image


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
promote
Staging environment
promote
Production environment


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
promote
Test environment
promote
Staging environment
promote
Production environment


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Production environment
promote
Staging environment
promote
Test environment
promote
Development environment
Continuous integration
Continuous delivery
Continuous deployment

Teamwork

Team Activities

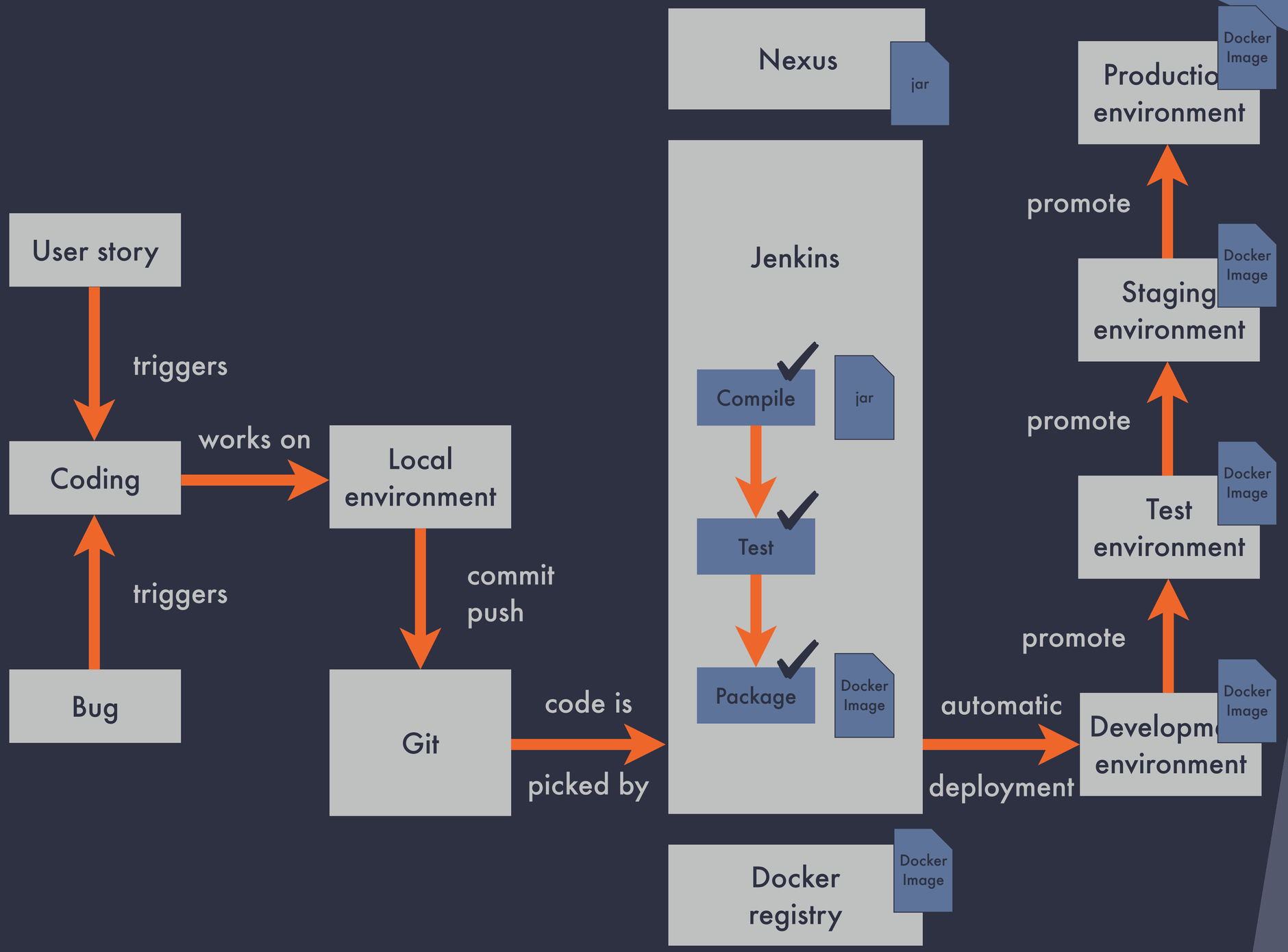
Scrum meetings

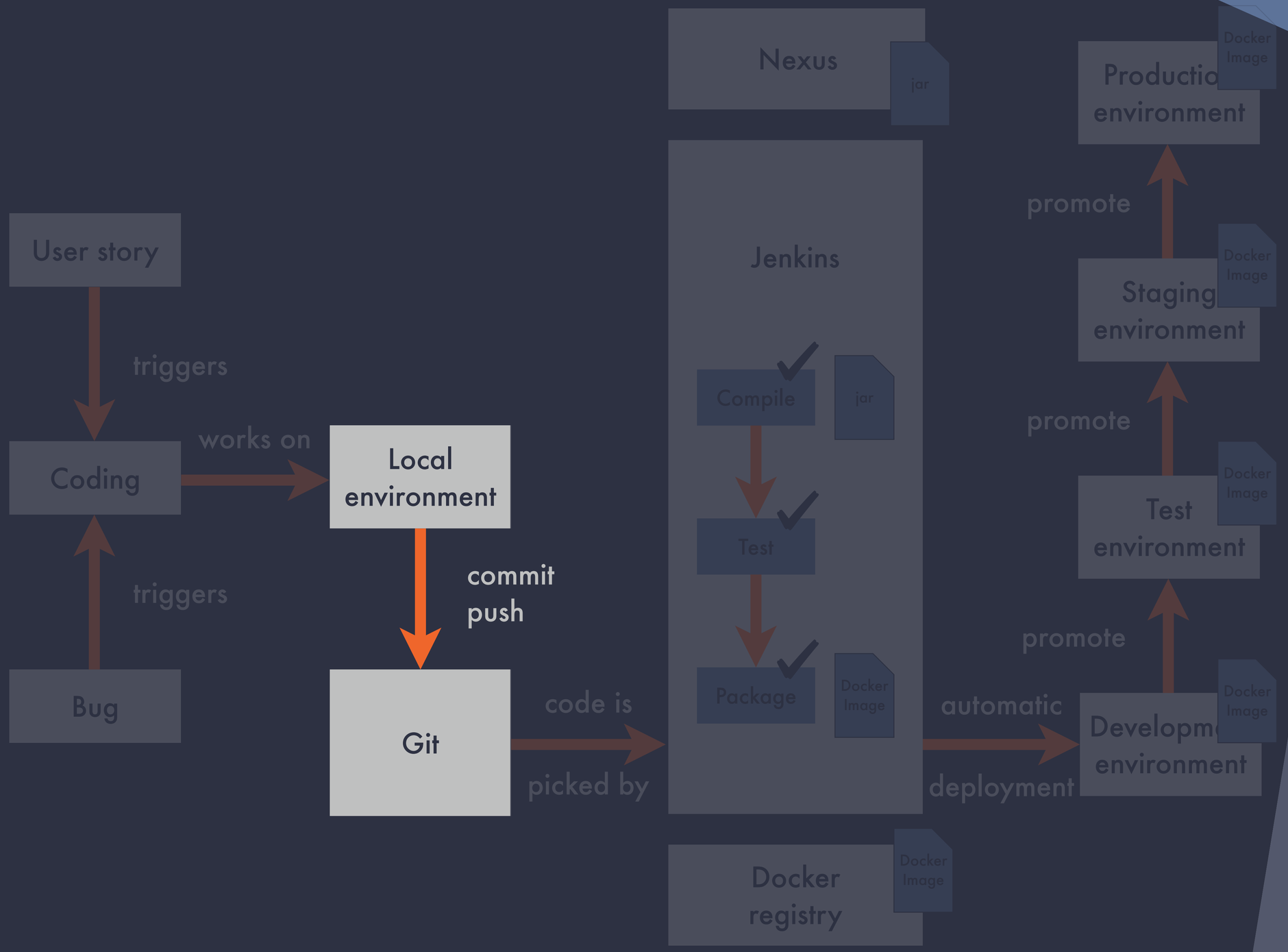
Whiteboard sessions

Ad hoc problem solving

Pair programming

Code reviews


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image


User story
triggers
Coding
works on
Local environment
commit
push
triggers
Bug
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image

Project knowledge

Project specific standards

Code Review

Code quality

Pair Programming

Find
Invoice
3
Alice

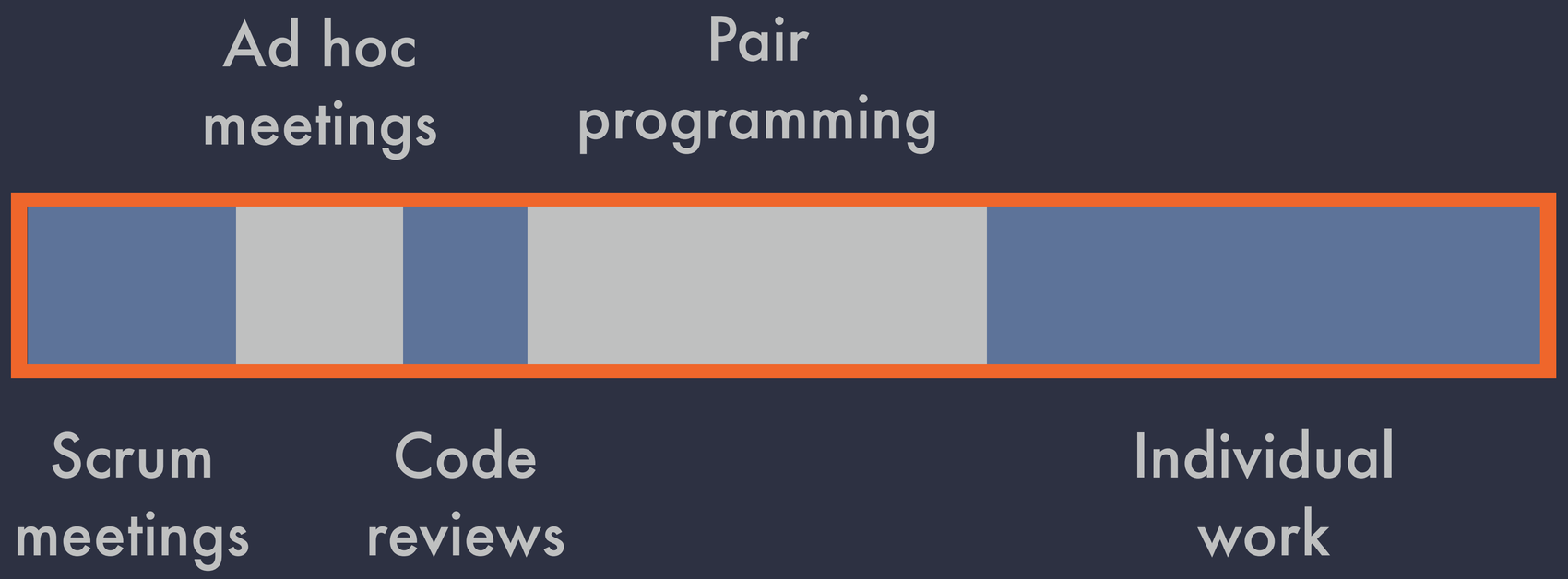
Find
Invoice
3
Alice

Ongoing code review

Knowledge spread across whole team

Not about tutoring

Individual Activities


Ad hoc meetings
Pair programming
Scrum meetings
Code reviews
Individual work

Ask questions, really!

Communication

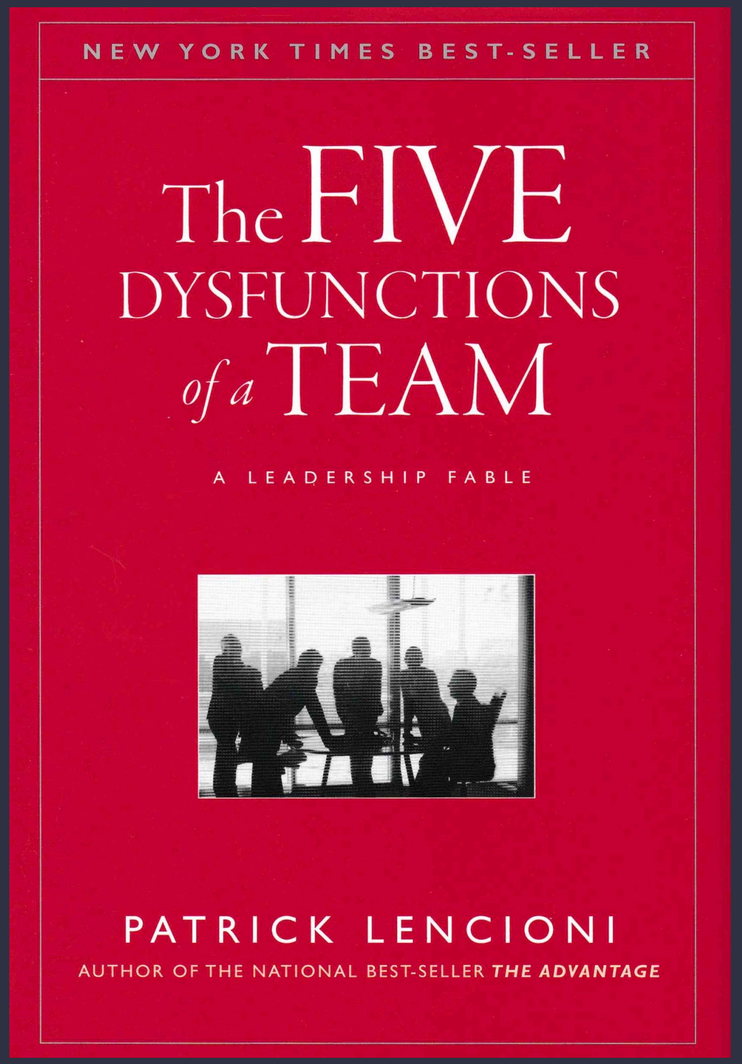
Ask questions, really!

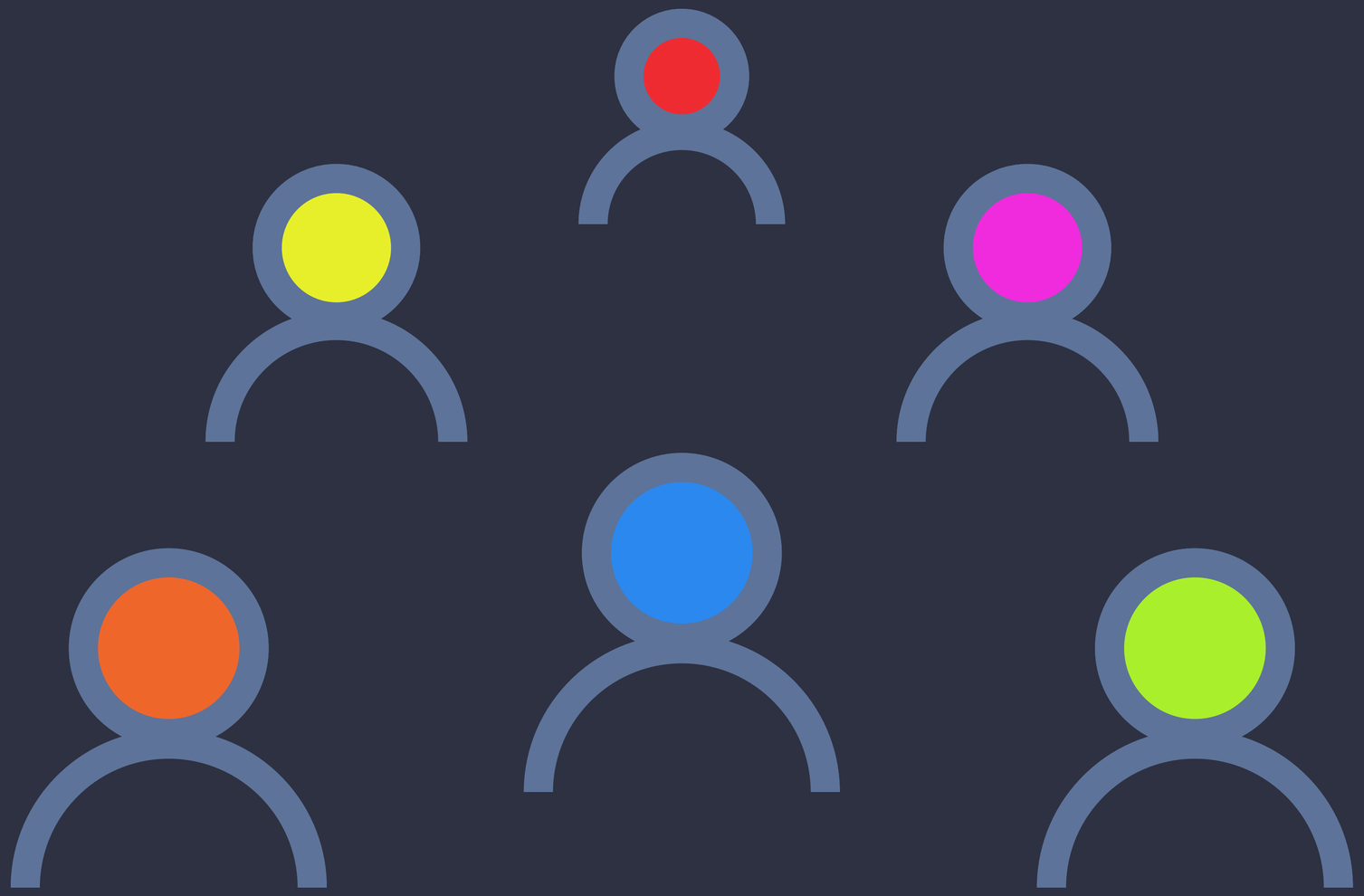
Communication

Suggest time boxed Q&A sessions.

Introverts ⟷ Extroverts

Effective Teamwork


NEW YORK TIMES BEST-SELLER
The FIVE DYSFUNCTIONS of a TEAM
A LEADERSHIP FABLE
PATRICK LENCIONI
AUTHOR OF THE NATIONAL BEST-SELLER THE ADVANTAGE



I don’t know...
I don’t understand...

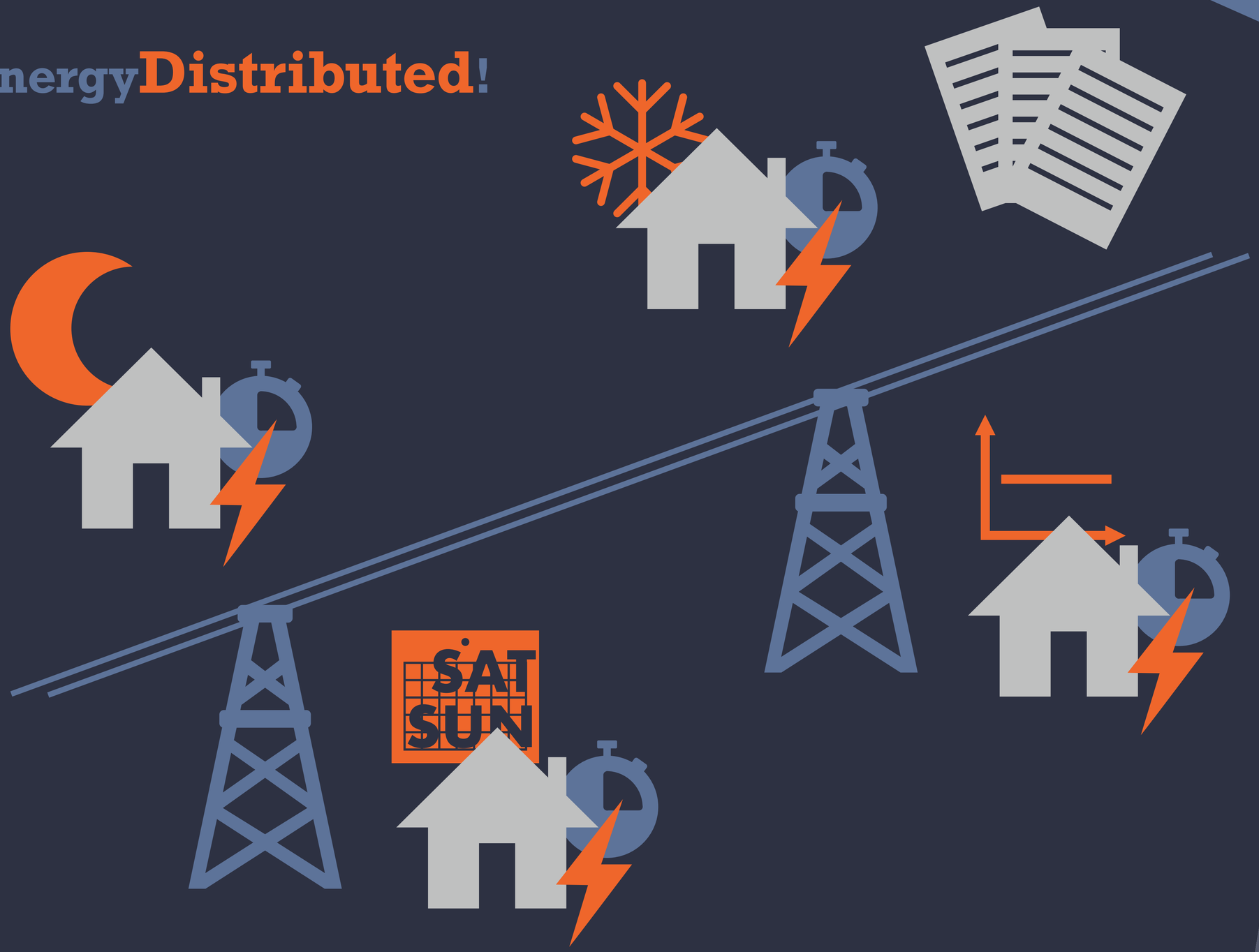
Summary

Module 3

The Project

A Business Case

EnergyDistributed!



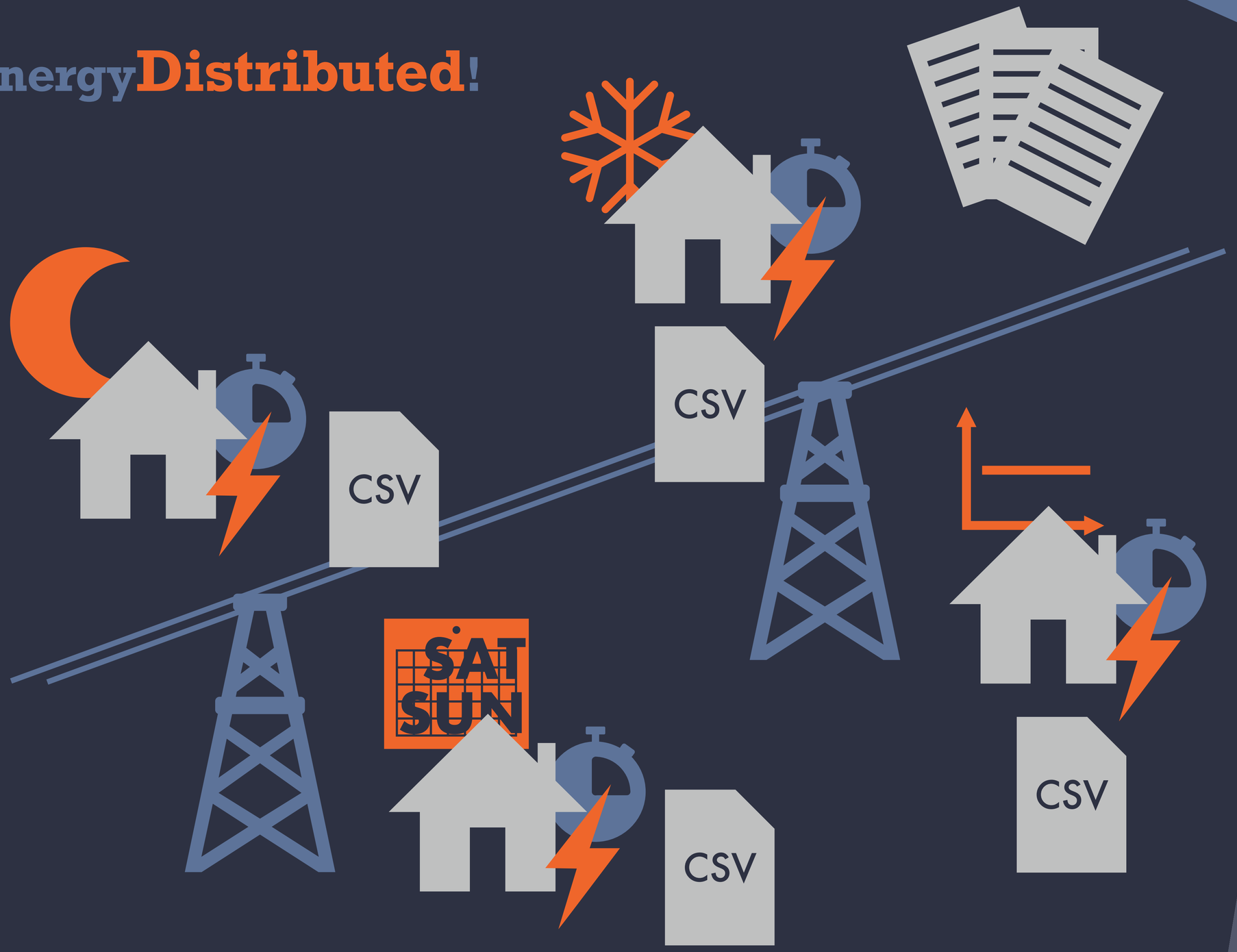
Project Name

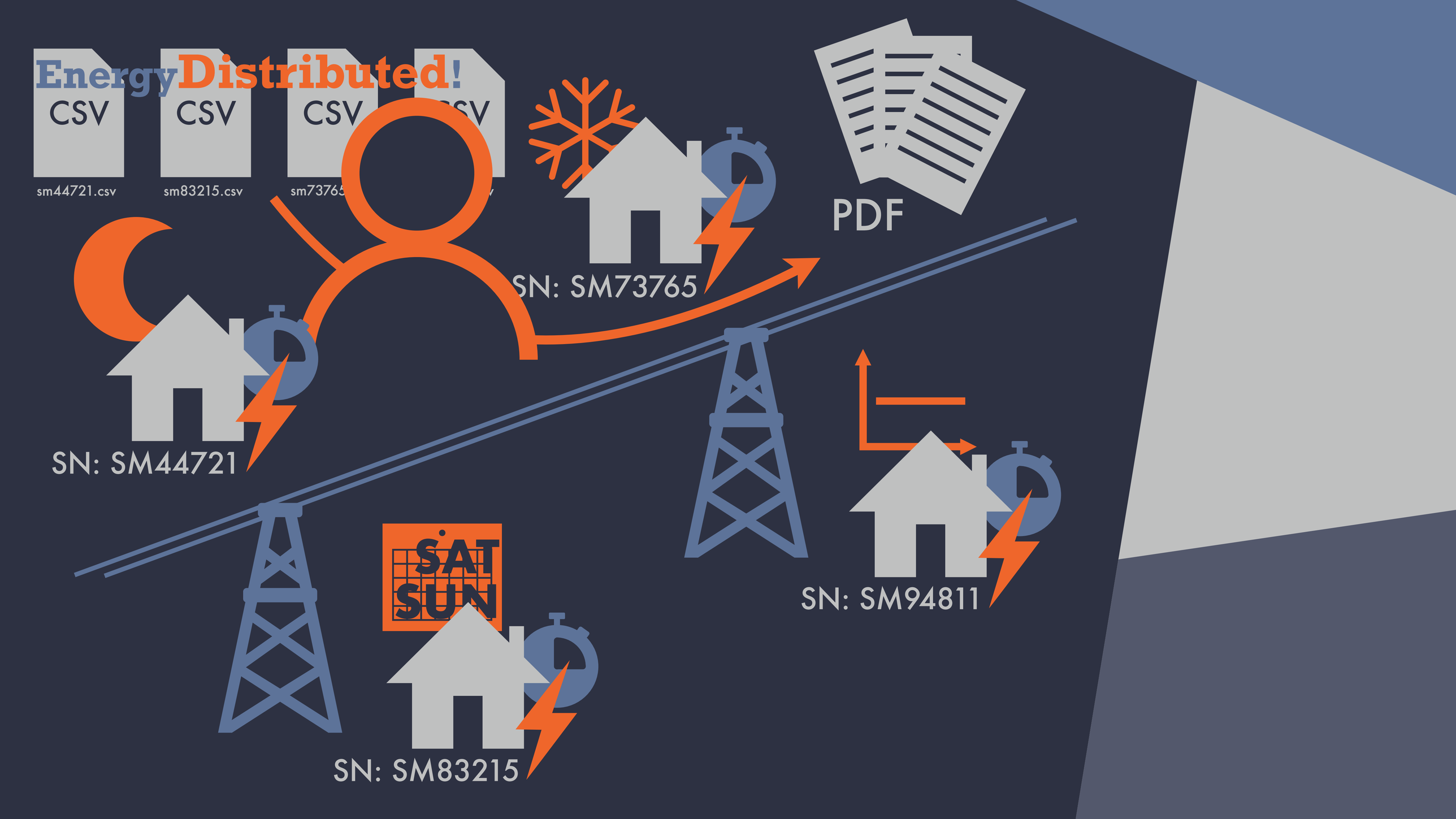
Energy Invoicer

NrgyInvoicR

More Details

EnergyDistributed!




Energy Distributed!
CSV
CSV
CSV
sm44721.csv
sm83215.csv
PDF
SN: SM73765
SN: SM44721
SAT SUN
SN: SM94811
SN: SM83215

EnergyDistributed!
CSV
CSV
CSV
sm44721.csv
sm83215.csv
PDF
SN: SM73765
SN: SM44721
SN: SM94811
SAT
SUN
SN: SM83215

User Stories

Define an energy plan.

Register a meter identified with a serial number.

Register a client with their name and address.

Assign a meter to a client.

Assign an energy plan to a client.

Import a readings file.

Generate an invoice.

Save generated invoice as a PDF file.

The Size of the Example

The Size of the Example

Small examples

Full-blown system

The Size of the Example

Small examples

Full-blown system

The Size of the Example

Small examples

Full-blown system

Disclaimer

Installation

Requirements

Requirements

Windows 10 Home

Intel i5 or similar
8GB RAM

General Purpose Tools

General Purpose Tools

Chrome

Notepad++

Docker

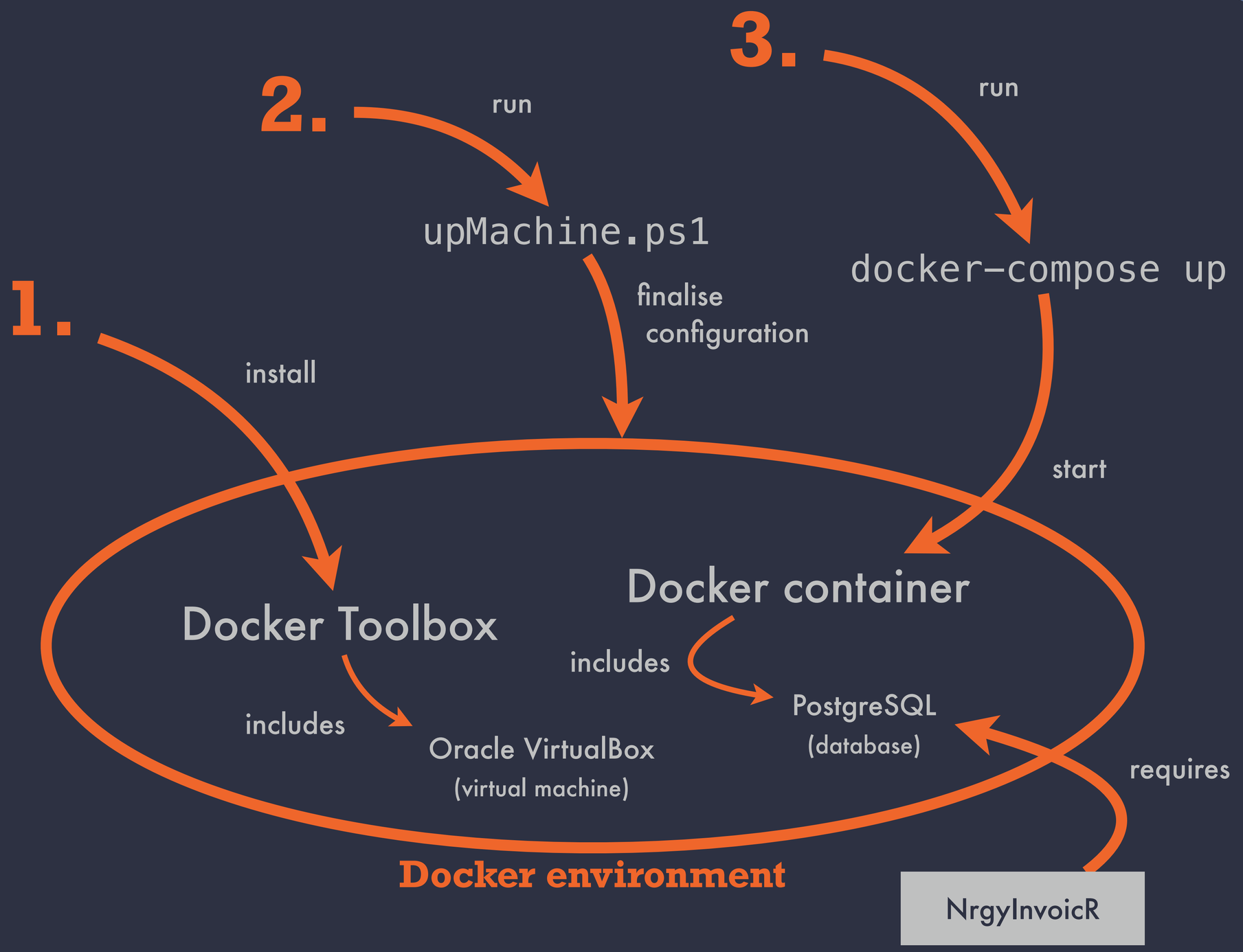
Docker

Docker Toolbox ✓

Docker Toolbox includes Git

Docker Desktop*

* if your system meets additional requirements


1.
install
2.
run
upMachine.ps1
finalise
configuration
3.
run
docker-compose up
start
Docker Toolbox
includes
Oracle VirtualBox
(virtual machine)
Docker container
includes
PostgreSQL
(database)
requires
NrgyInvoicR
Docker environment

Cloning Git Project

Cloning Git Project

http://github.com/bkaminnski/NrgyInvoicR

IDE

IDE

Integrated Development Environment

IntelliJ Community Edition

Running the Project

Supporting Tools

Supporting Tools

pgAdmin

Jaspersoft Studio

Java in Console

Java in Console

JVM

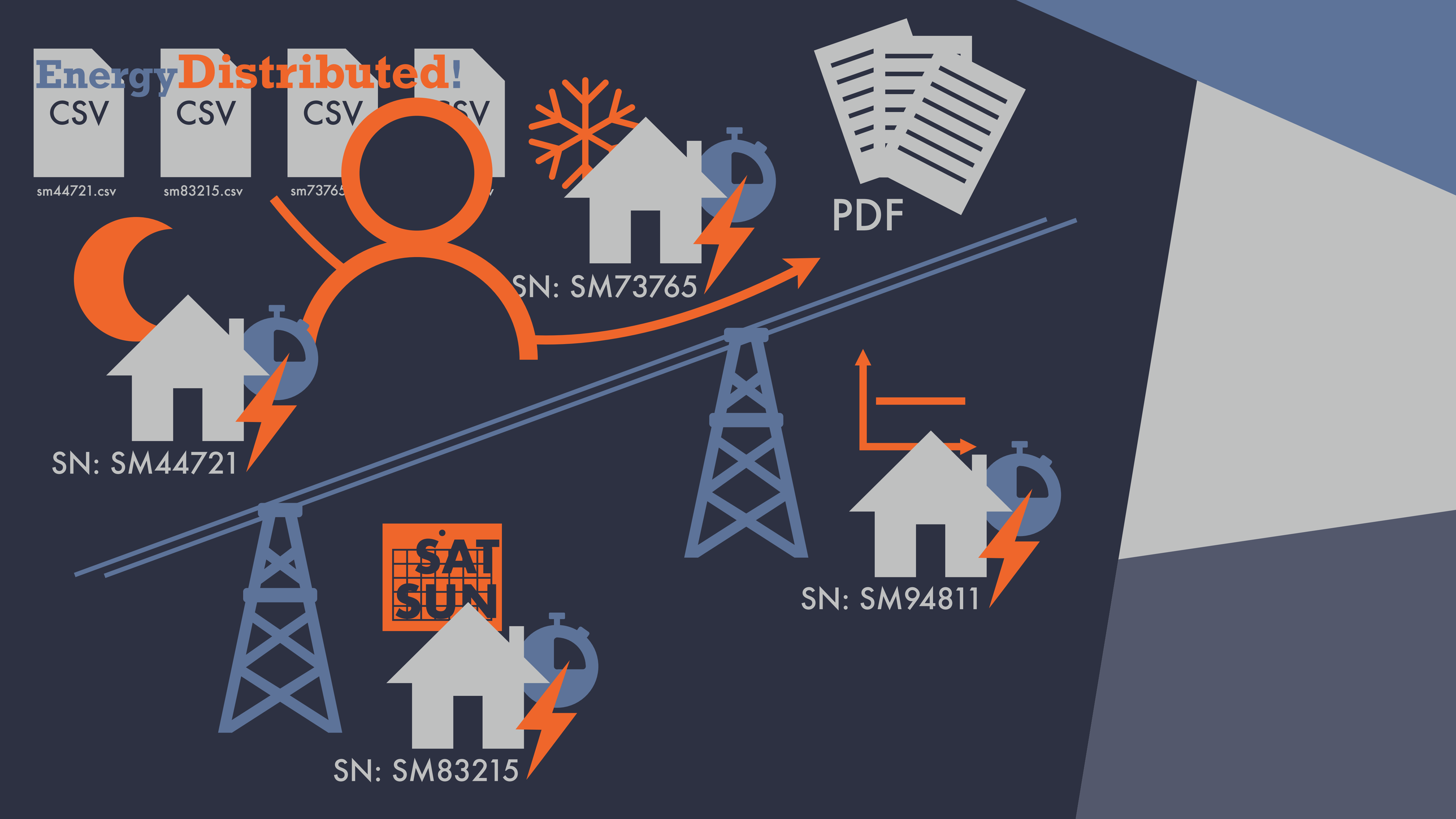
Maven

Frontend Development

Frontend Development

Node.js Angular CLI Visual Studio Code

Demo


Energy Distributed!
CSV
CSV
CSV
sm44721.csv
sm83215.csv
PDF
SN: SM73765
SN: SM44721
SAT SUN
SN: SM94811
SN: SM83215

Define an energy plan.

Register a meter identified with a serial number.

Register a client with their name and address.

Assign a meter to a client.

Assign an energy plan to a client.

Import a readings file.

Generate an invoice.

Save generated invoice as a PDF file.

Define an energy plan.

Register a meter identified with a serial number.

Register a client with their name and address.

Assign a meter to a client.

Assign an energy plan to a client.

Import a readings file.

Generate an invoice.

Save generated invoice as a PDF file.

Define an energy plan.

Register a meter identified with a serial number.

Register a client with their name and address.

Assign a meter to a client.

Assign an energy plan to a client.

Import a readings file.

Generate an invoice.

Save generated invoice as a PDF file.

Define an energy plan.

Register a meter identified with a serial number.

Register a client with their name and address.

Assign a meter to a client.

Assign an energy plan to a client.

Import a readings file.

Generate an invoice.

Save generated invoice as a PDF file.

Define an energy plan.

Register a meter identified with a serial number.

Register a client with their name and address.

Assign a meter to a client.

Assign an energy plan to a client.

Import a readings file.

Generate an invoice.

Save generated invoice as a PDF file.

Summary

Module 4

Basics of Software Development

Data Formats, Java, Spring,
Frontend Development, Git, Databases,
Networks and Infrastructure

Data Formats

Data Formats

Binary

Readable only by machines

Text-based

Readable by both machines and humans

XML, JSON, YAML, CSV

XML

XML

Extensible Markup Language

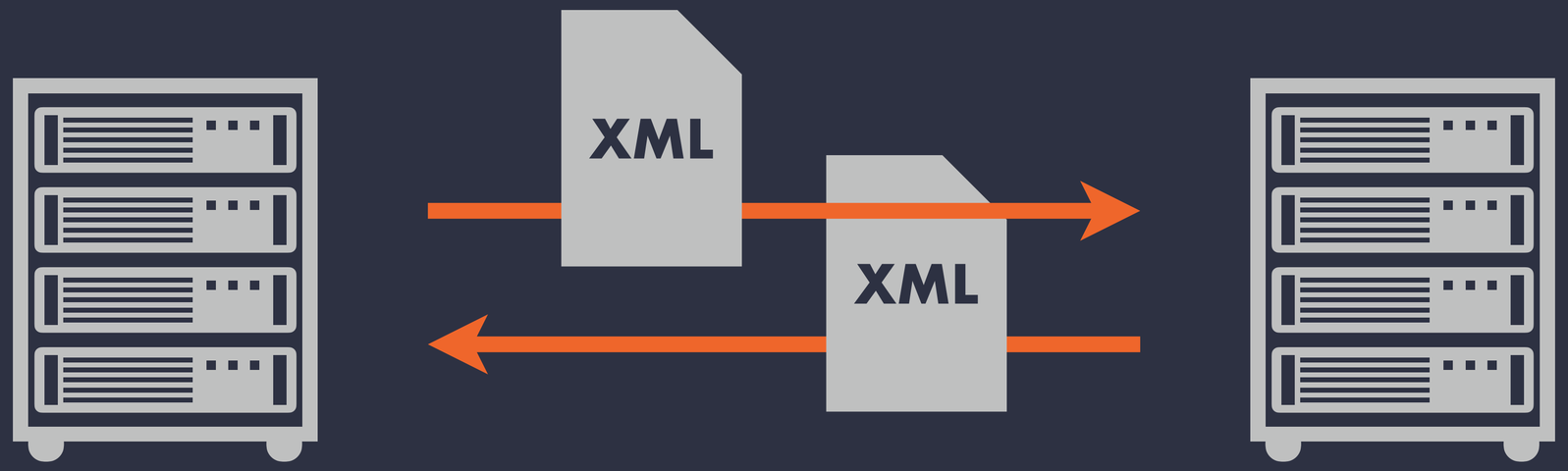
```xml
<?xml version="1.0" encoding="UTF-8"?>
<meter serialNumber="9bce2edb-d4b9-401e-9f6d-eec59f0e7f45">
  <reading>
    <date>2019-02-08</date>
    <values>
      <value date="2019-01-01T00:00+01:00">0.0471</value>
      <value date="2019-01-02T00:00+01:00">0.0152</value>
      <value date="2019-01-03T00:00+01:00">0.0201</value>
      <value date="2019-01-04T00:00+01:00">0.0197</value>
      <value date="2019-01-05T00:00+01:00">0.0132</value>
    </values>
  </reading>
</meter>
```

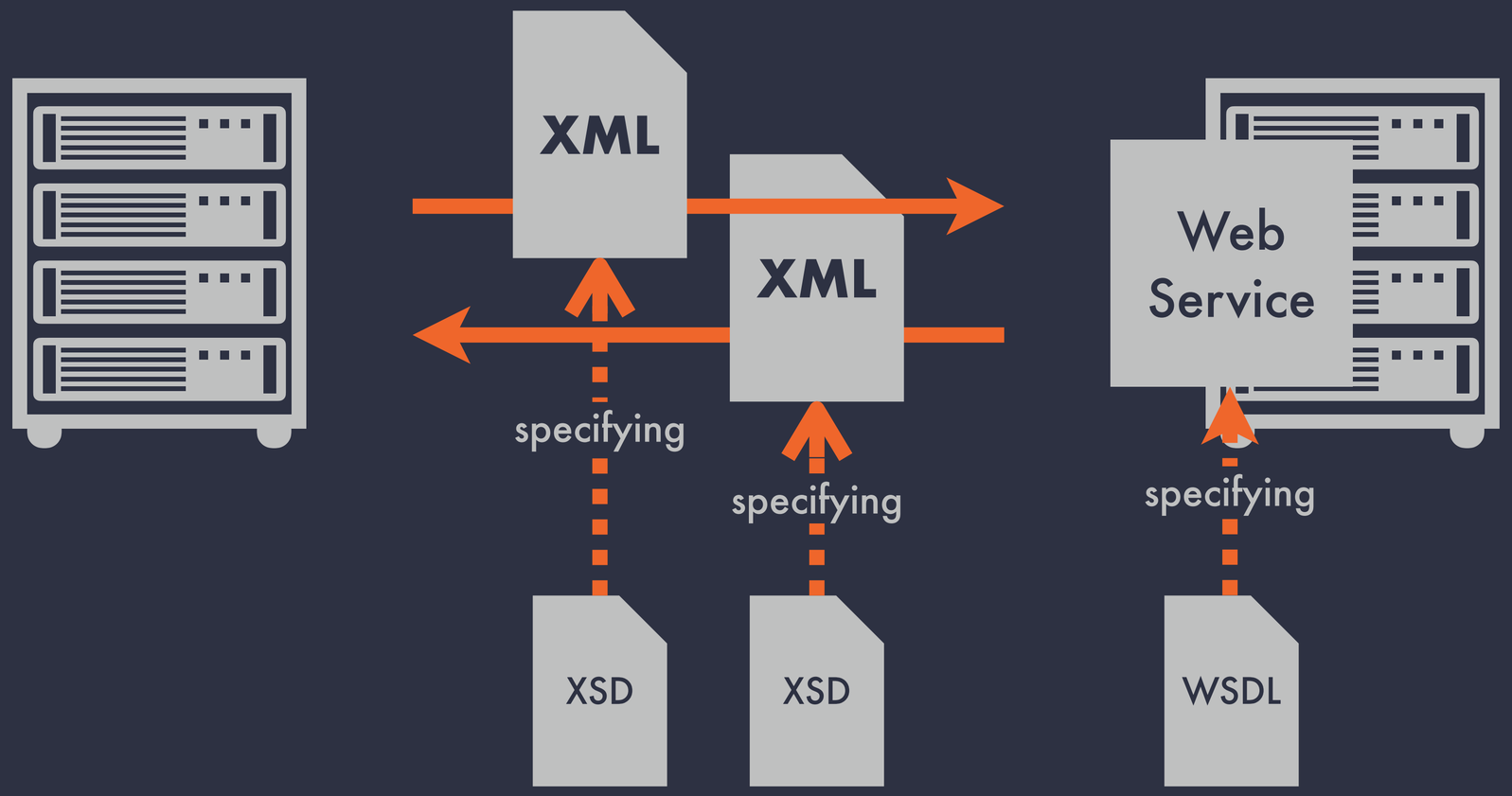
```
<?xml version="1.0" encoding="UTF-8"?>
<meter serialNumber="9bce2edb-d4b9-401e-9f6d-eec5
  <reading>
    <date>2019-02-08</date>
    <values>
      <value date="2019-01-01T00:0
      <value date="2019-01-02T
      <value date="2019-01
      <value date="201
      <value date="
    </values>
  </readin
</mete
```

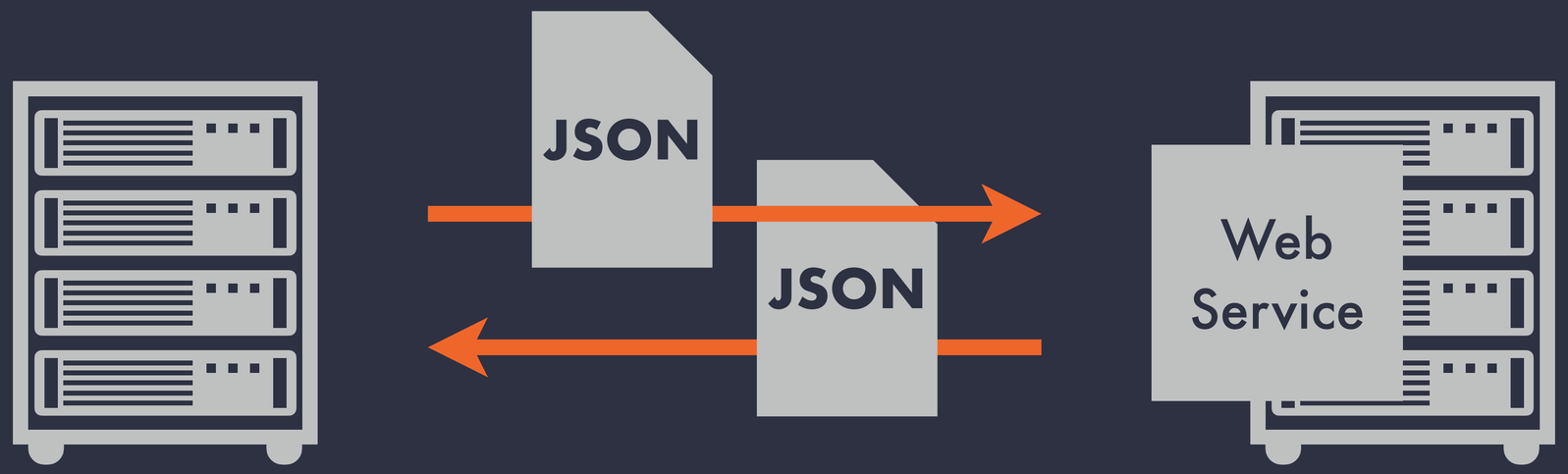
XML structure can be described by

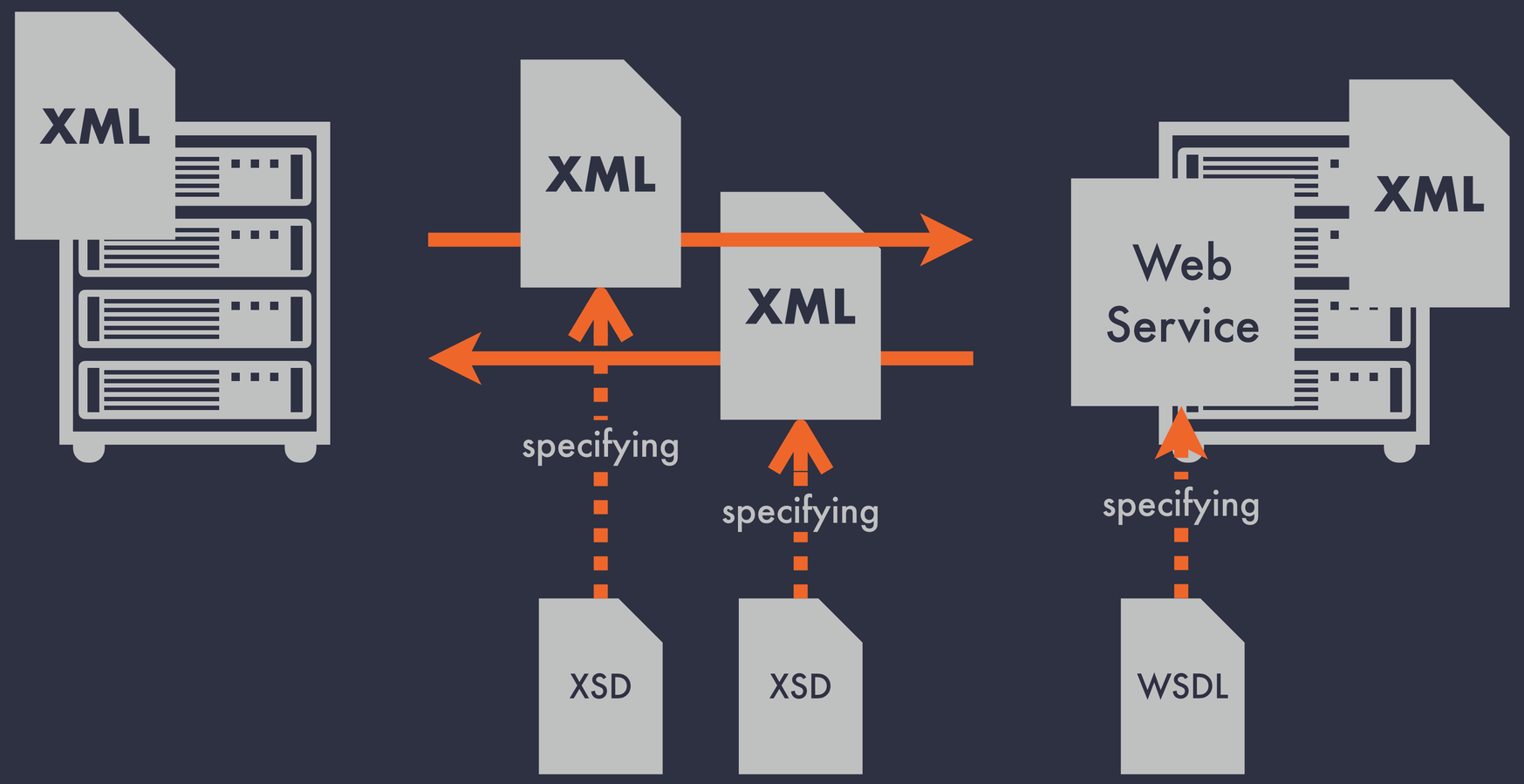
XSD

XML Schema Definition


XML
XML


XML
XML
Web Service
specifying
specifying
specifying
XSD
XSD
WSDL


JSON
JSON
Web
Service


XML
XML
XML
specifying
specifying
XSD
XSD
Web
Service
XML
specifying
WSDL

XML

XMLHTML

HTML

HyperText Markup Language

```html
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>History of WWW</title>
  </head>
  <body>
    <p>
      This <a href="https://home.cern/science/computing/birth-web/short-history-web">web page</a>
      describes the birth of the <b>World Wide Web<b>.
    </p>
  </body>
<html>
```

Beginning of WWW

Tim Berners-Lee, 1989

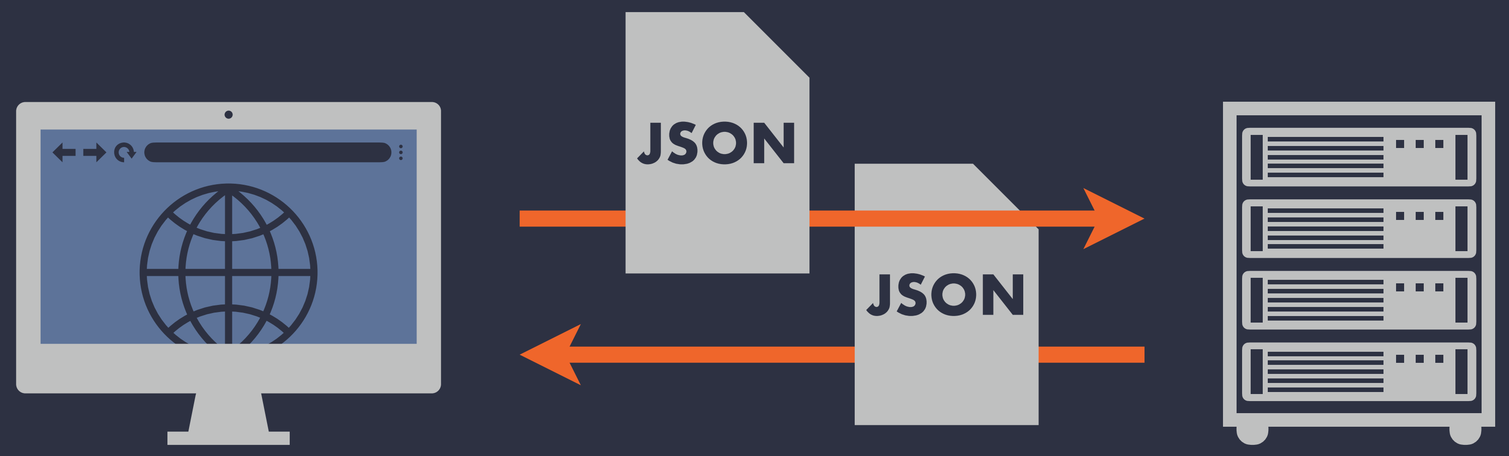
https://home.cern/science/computing/birth-web/short-history-web

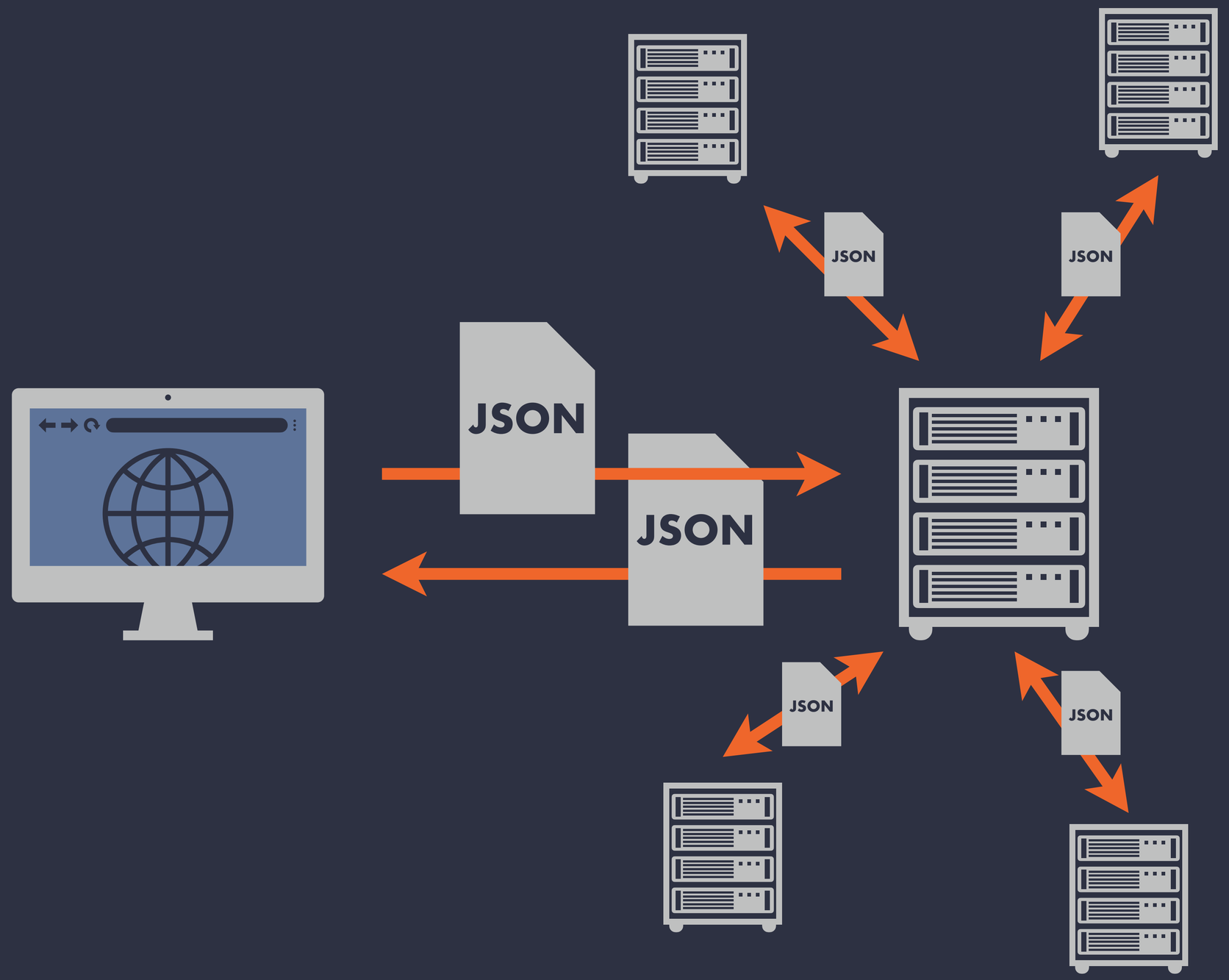
JSON

JSON

JavaScript Object Notation

```
{
   "serialNumber": "9bce2edb-d4b9-401e-9f6d-eec59f0e7f45",
   "reading": {
      "date": "2019-02-08",
      "values": [
         {
            "date": "2019-01-01T00:00+01:00",
            "value": "0.0471"
         },
         {
            "date": "2019-01-02T00:00+01:00",
            "value": "0.0152"
         },
         {
            "date": "2019-01-03T00:00+01:00",
            "value": "0.0201"
         },
         {
            "date": "2019-01-04T00:00+01:00",
            "value": "0.0197"
         },
         {
            "date": "2019-01-05T00:00+01:00",
            "value": "0.0132"
         }
      ]
   }
}
```


JSON
JSON


JSON
JSON
JSON
JSON
JSON
JSON

YAML

YAML

YAML Ain’t Markup Language

XML	JSON	YAML
Documents	Data	Configuration
Data	Configuration	
Configuration		

CSV

Microsoft Excel
CSV
Google Sheets

CSV

Comma Separated Values

```
2019-01-01T00:00+01:00;0.0471
2019-01-02T00:00+01:00;0.0152
2019-01-03T00:00+01:00 0.0201
2019-01-04T00:00+01:00;0.0197
2019-01-05T00:00+01:00 0.0132
```

Meter Reading

2019-01-01T00:00+01:00	0.0471
2019-01-02T00:00+01:00	0.0152
2019-01-03T00:00+01:00	0.0201
2019-01-04T00:00+01:00	0.0197
2019-01-05T00:00+01:00	0.0132

```
2019-01-01T00:00+01:00;0.0471
2019-01-02T00:00+01:00;0.0152
2019-01-03T00:00+01:00 0.0201
2019-01-04T00:00+01:00;0.0197
2019-01-05T00:00+01:00 0.0132
```

Meter Reading

2019-01-01T00:00+01:00	0.0471
2019-01-02T00:00+01:00	0.0152
2019-01-03T00:00+01:00	0.0201
2019-01-04T00:00+01:00	0.0197
2019-01-05T00:00+01:00	0.0132

Java

JVM, JDK and JRE, Memory, Maven, Programming Tasks, Unit Testing, Debugging, Tips on Learning Programming

JVM, JDK and JRE

JVM

JVM

Java Virtual Machine

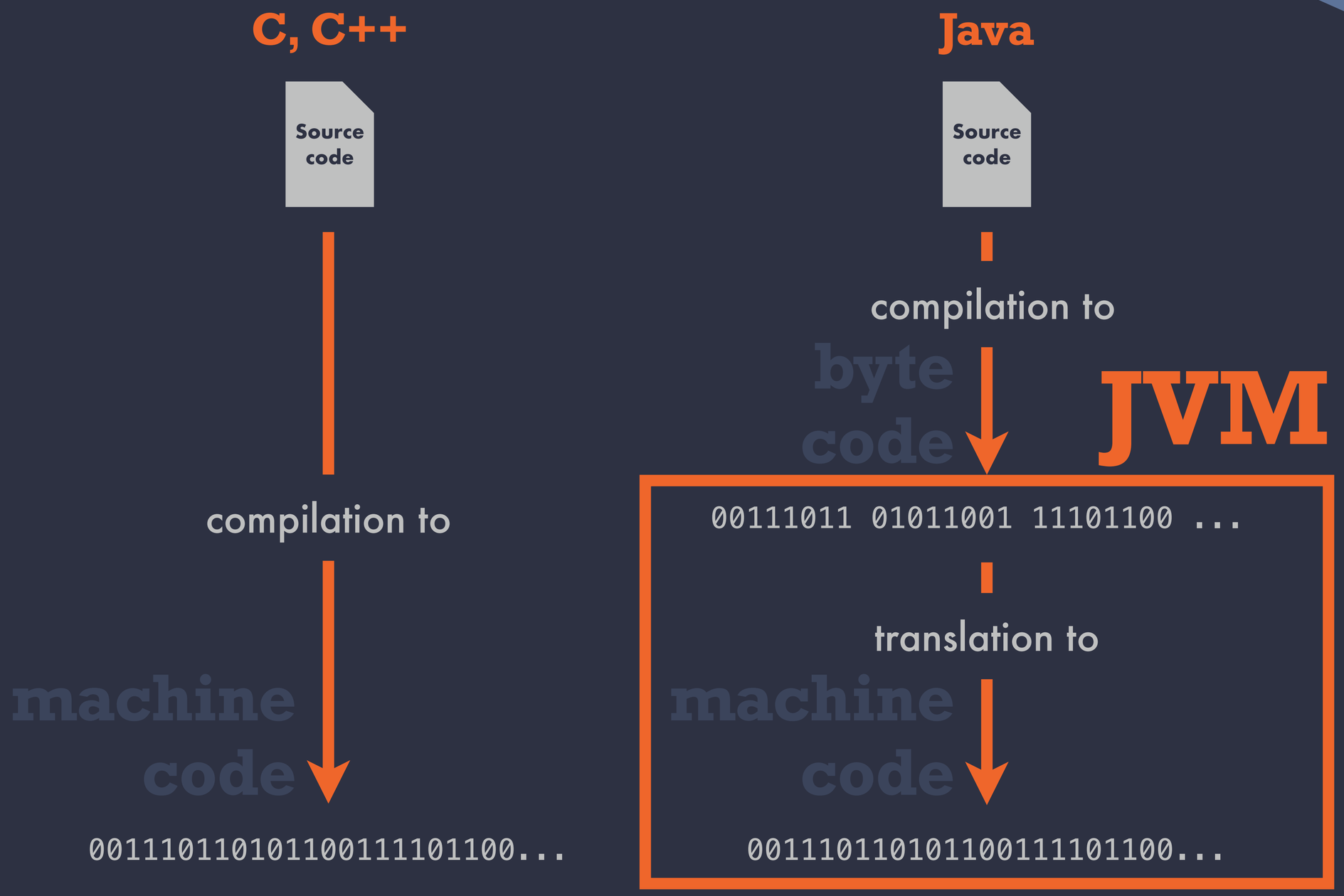
C, C++

compilation to

machine
code

```
00111011010110011110110 0...
```

Application.exe in Windows operating system


C, C++
Source code
compilation to
machine code
00111011010110011110110 0...
Java
Source code
compilation to
byte code
JVM
00111011 01011001 11101100 ...
translation to
machine code
00111011010110011110110 0...

JRE
java.exe

Java Runtime Environment

```
C:\> java -jar application.jar
```

starts

JVM

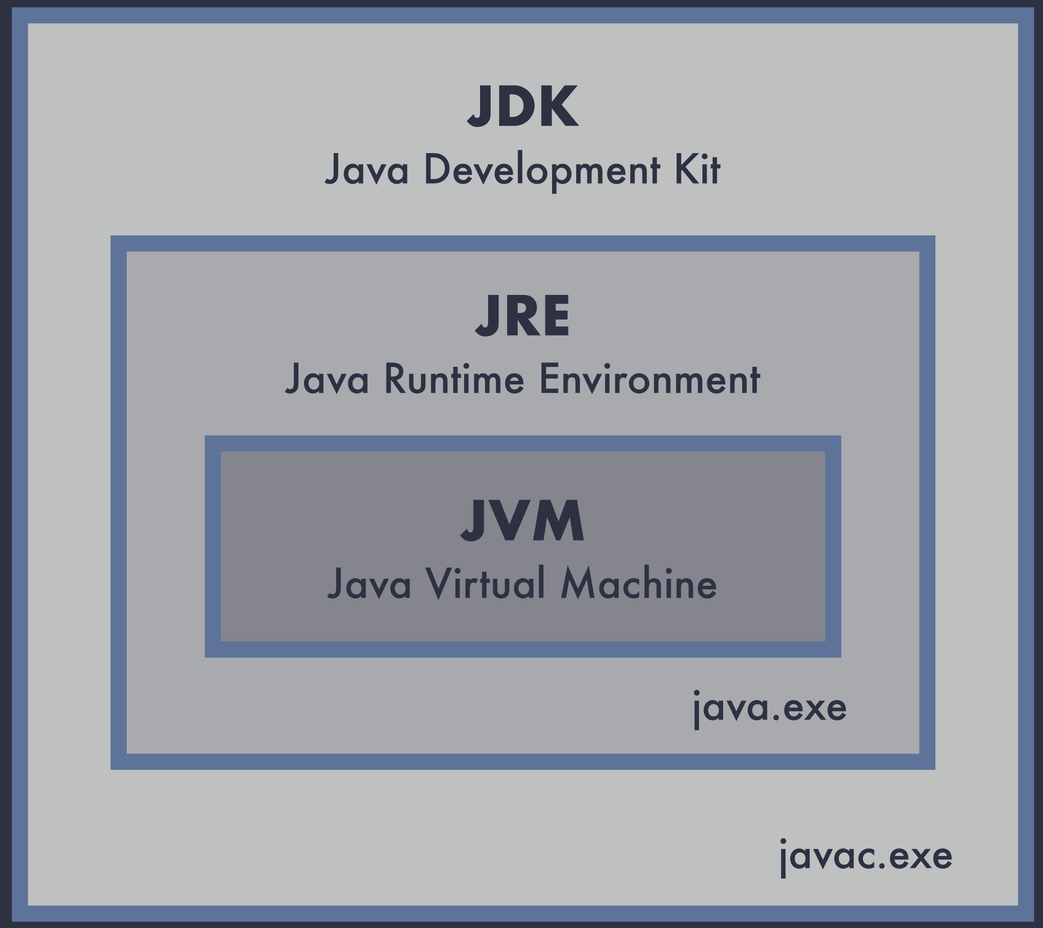
application.jar

JDK

Java Development Kit

javac.exe

JDK
Java Development Kit
JRE
Java Runtime Environment
java.exe
javac.exe


JDK
Java Development Kit
JRE
Java Runtime Environment
JVM
Java Virtual Machine
java.exe
javac.exe

Memory

```
int a = 2;
int b = a + 1;
```

RAM
Random-Access
Memory

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```

Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```

Stack

Heap

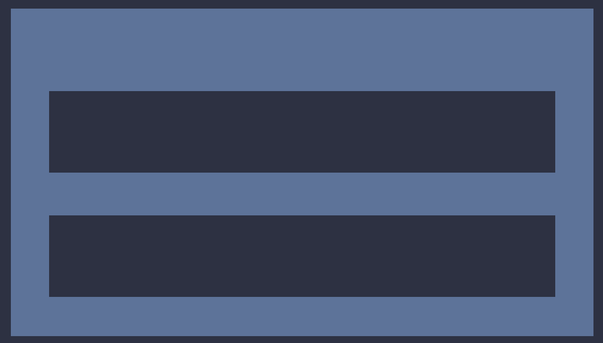
Local variable of a primitive data type

Creation of a new object

A reference to an object

A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```



Stack

Heap

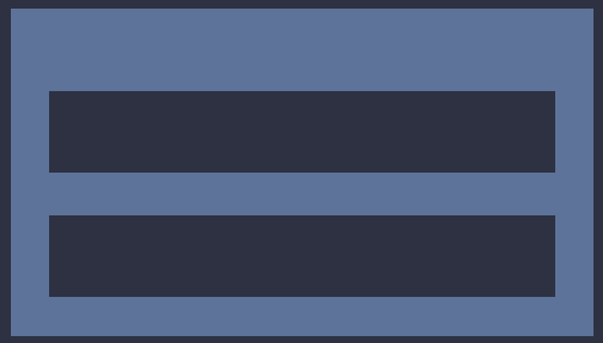
Local variable of a primitive data type

Creation of a new object

A reference to an object

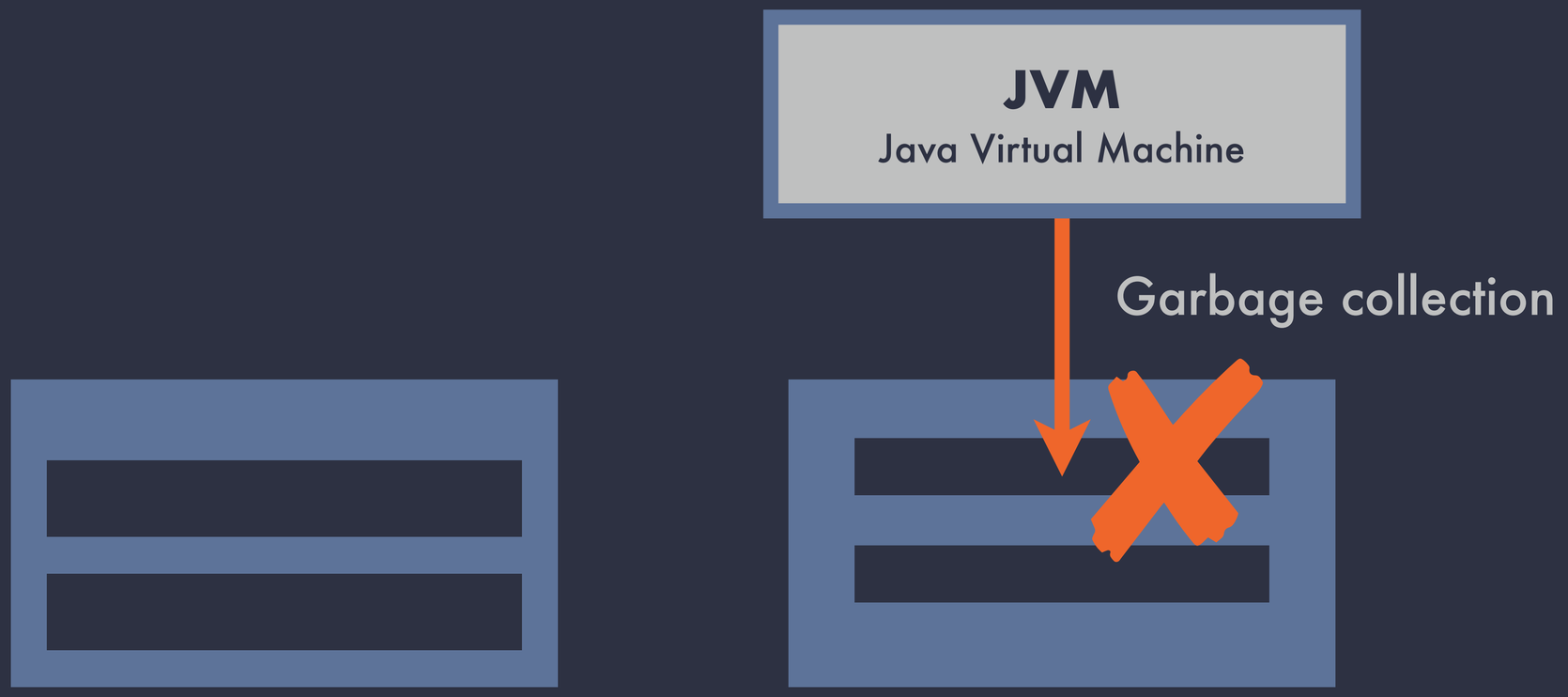
A method call

```
public List<ReadingValue> generateReadingValues() {
    List<ReadingValue> readingValues = new ArrayList<>();
    for (int i = 0; i < 10; i++) {
        ReadingValue readingValue = generateReadingValue();
        readingValues.add(readingValue);
    }
    return readingValues;
}

private ReadingValue generateReadingValue() {
    return new ReadingValue();
}
```



Stack

Heap



Stack

Static memory

Heap

Managed memory

StackOverflowError

Stack trace

OutOfMemoryError

Maven

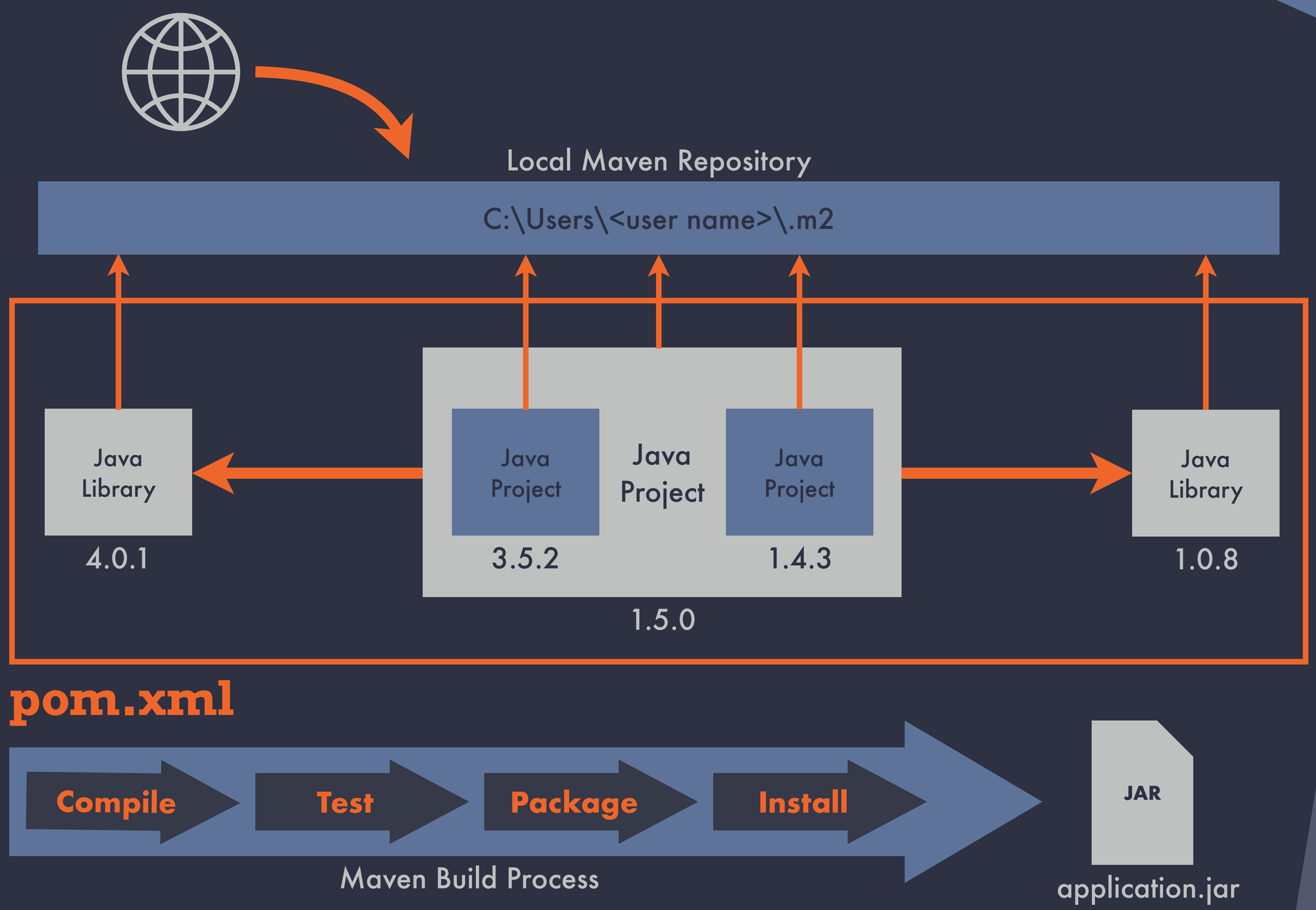
Maven

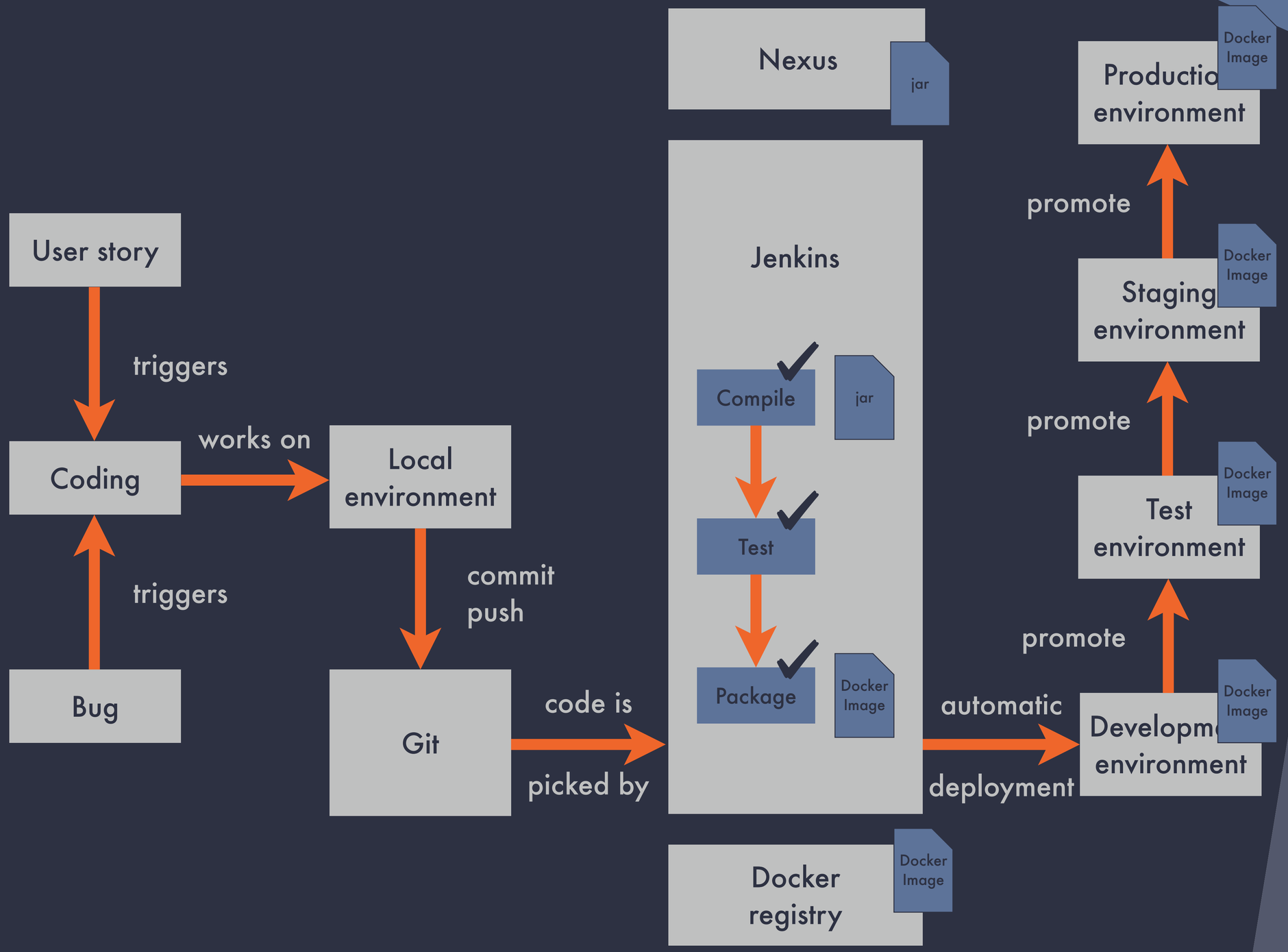
Project management tool

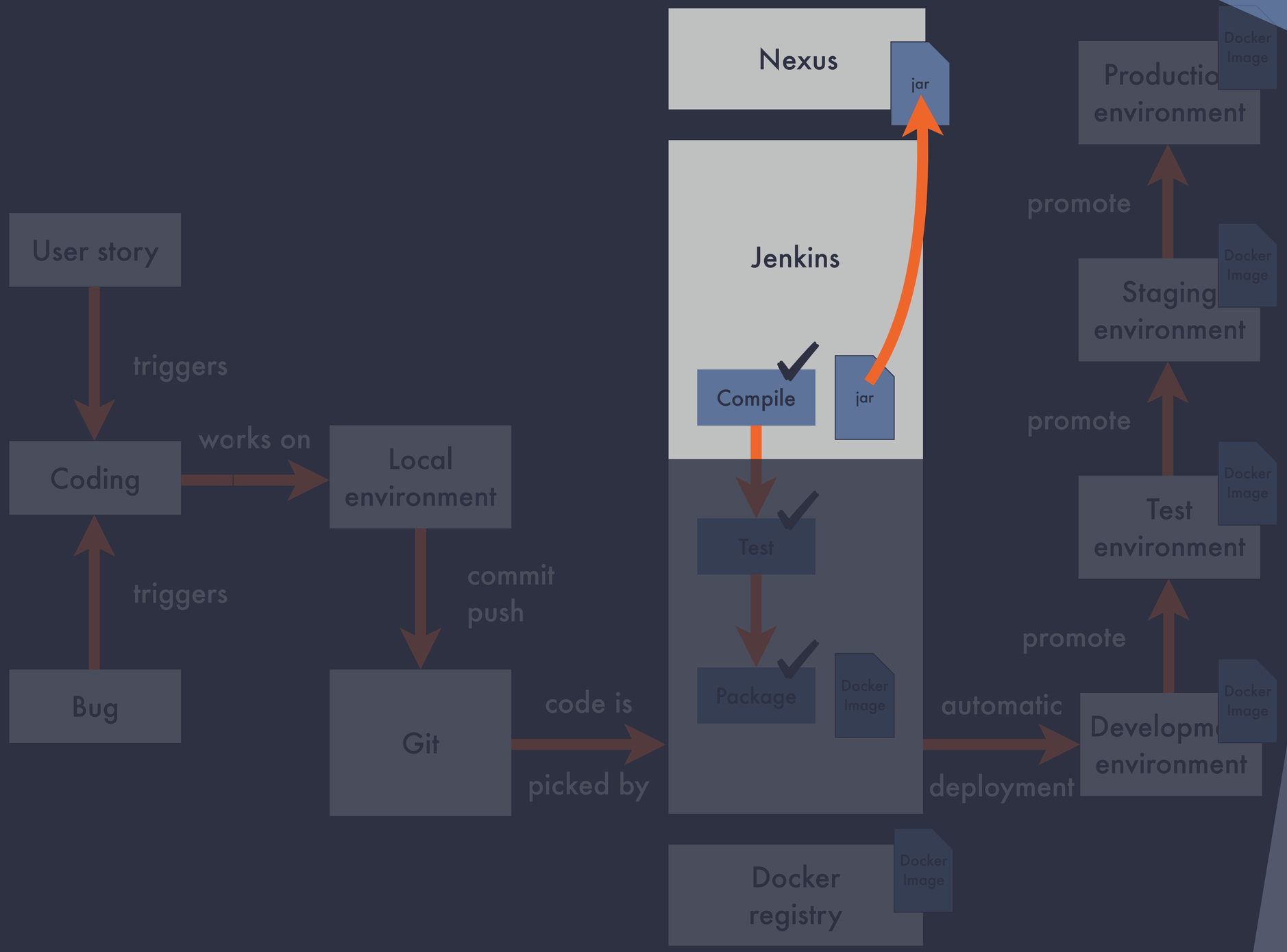
Maven

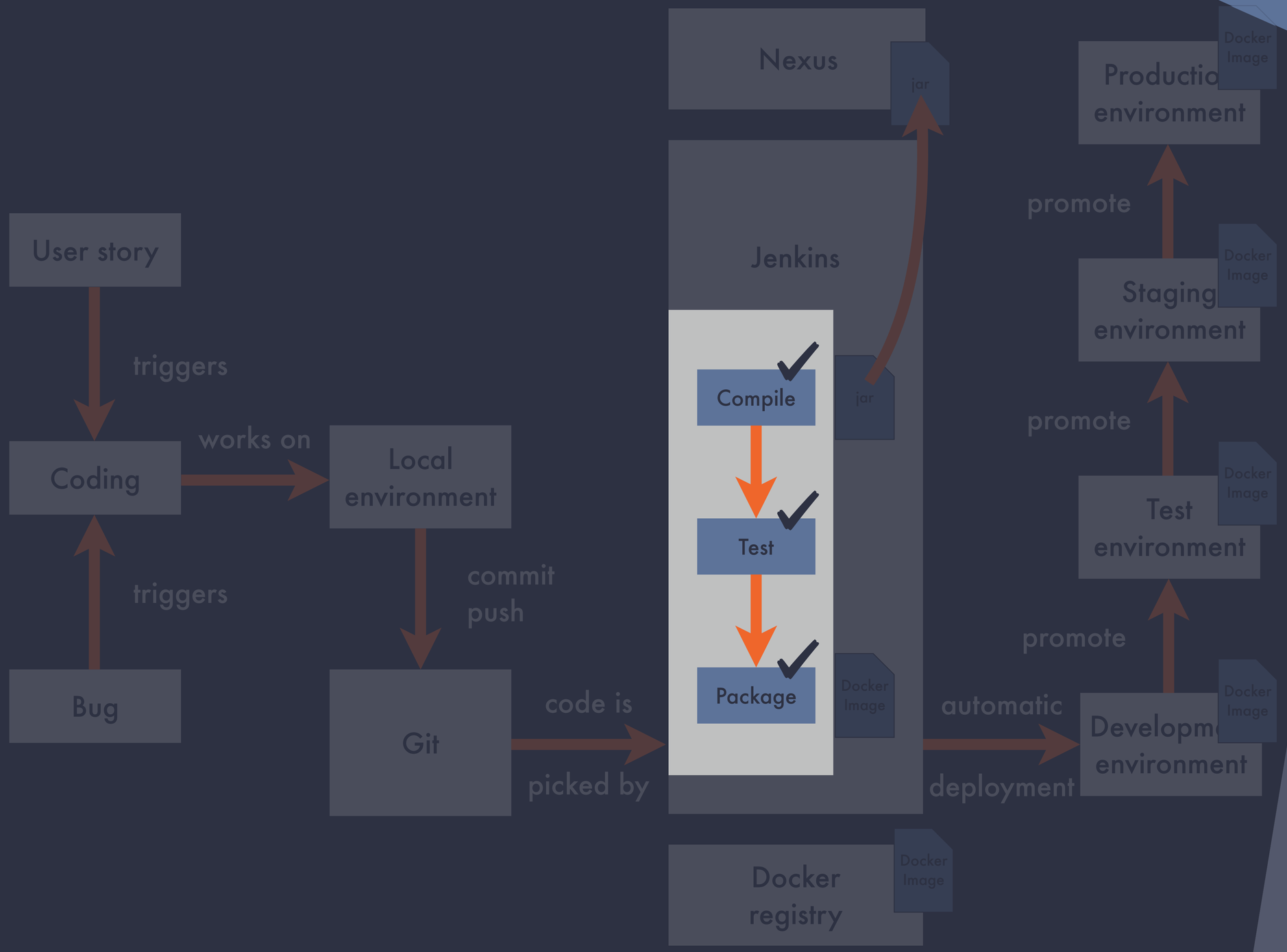
Java project management tool

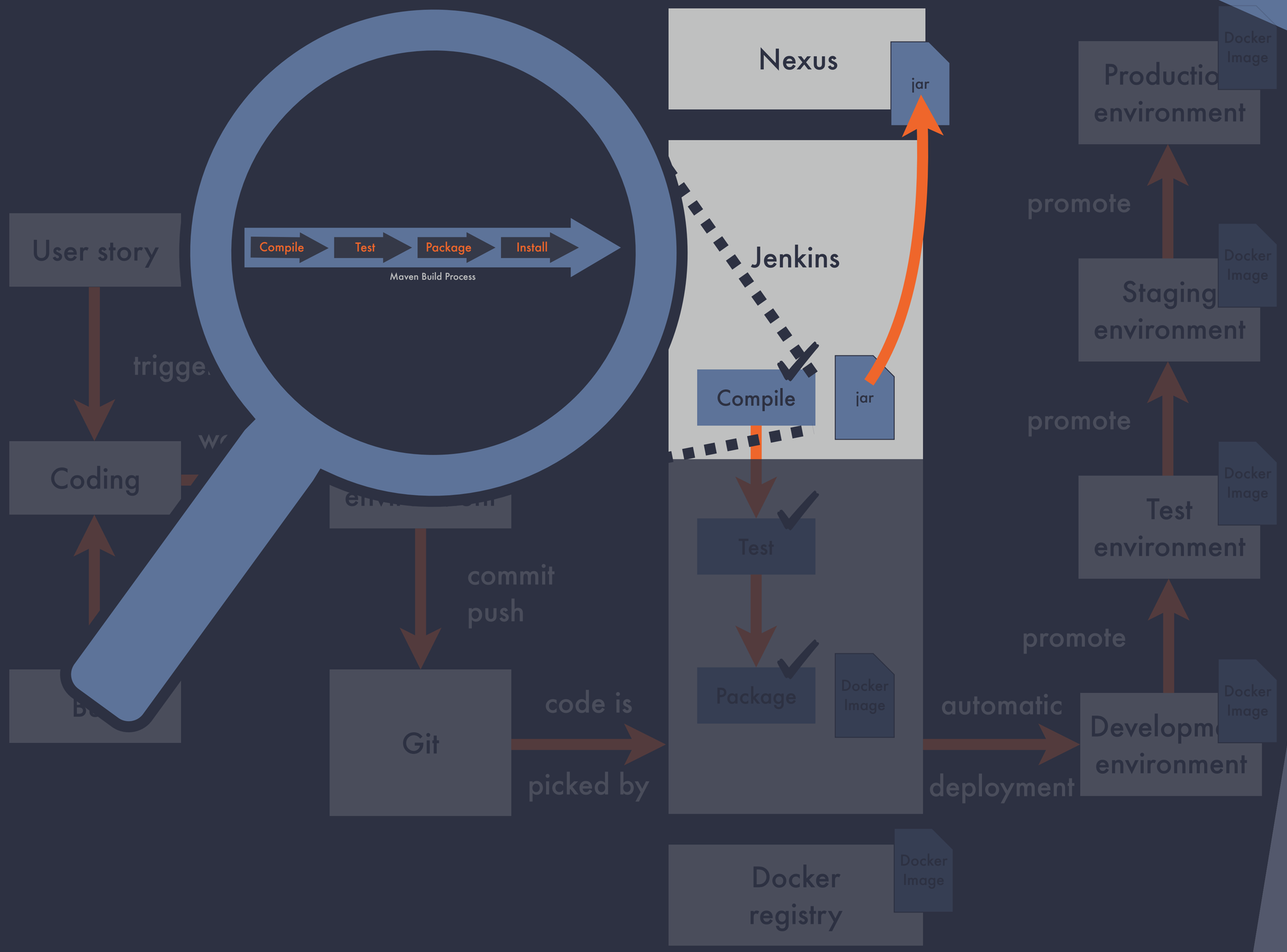
Java
Library
Java
Project
Java
Library


Local Maven Repository
C:\Users\<user name>\.m2
Java Library
4.0.1
Java Project
3.5.2
Java Project
Java Project
1.4.3
1.5.0
Java Library
1.0.8
pom.xml
Compile
Test
Package
Install
Maven Build Process
JAR
application.jar

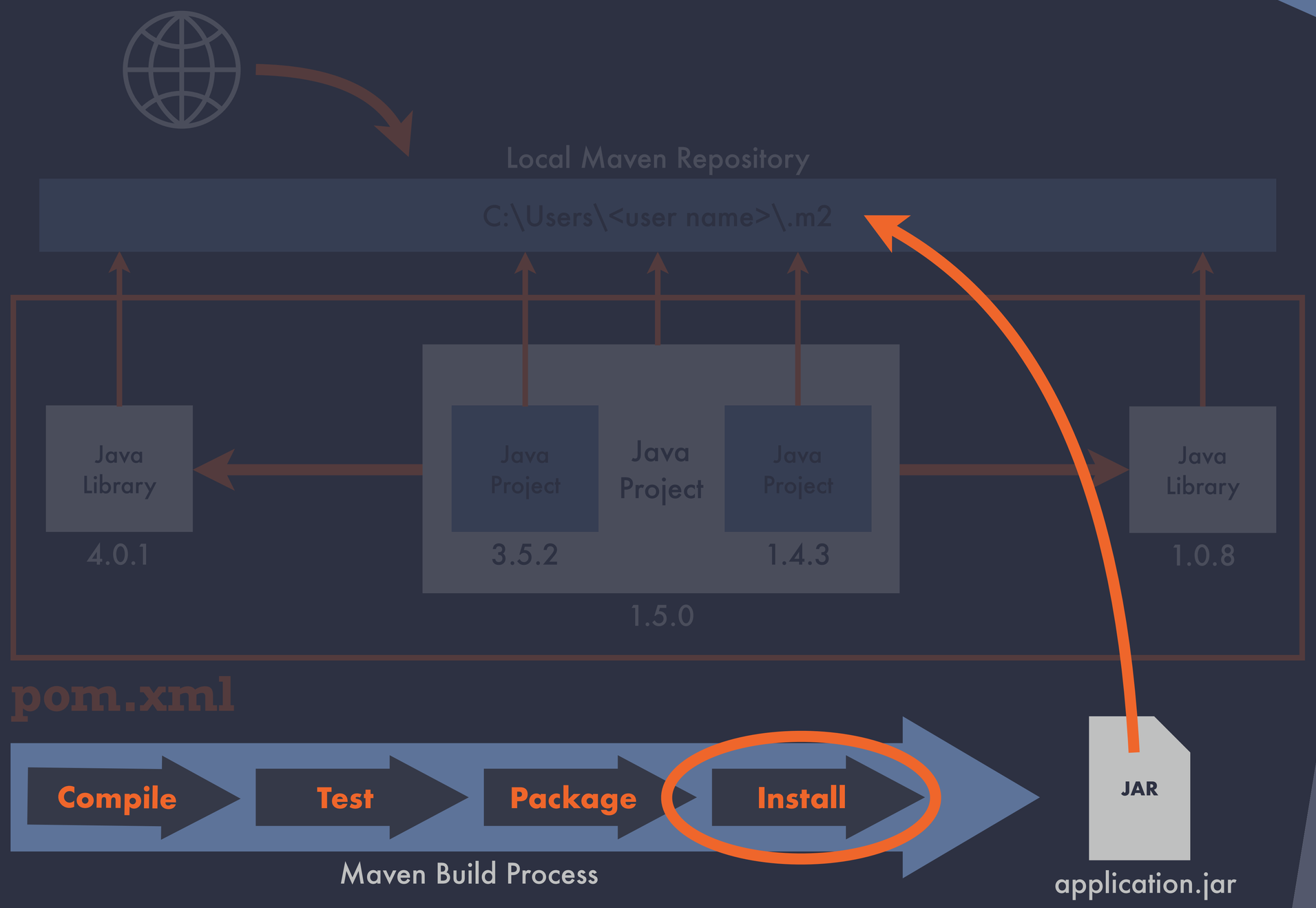

User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image


User story
triggers
Coding
works on
Local environment
triggers
commit push
Bug
Git
code is picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
Test
Package
jar
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image


Compile
Test
Package
Install
Maven Build Process
Nexus
jar
Jenkins
Compile
jar
User story
Coding
commit
push
Git
code is
picked by
Test
Package
Docker
Image
Docker
registry
Docker
Image
automatic
deployment
Development
environment
promote
Test
environment
promote
Staging
environment
promote
Production
environment

```
C:\> mvn clean

C:\> mvn clean install

C:\> mvn install
```


Local Maven Repository
C:\Users\<user name>\.m2
Java Library
4.0.1
Java Project
3.5.2
Java Project
Java Project
1.4.3
1.5.0
Java Library
1.0.8
pom.xml
Compile
Test
Package
Install
Maven Build Process
JAR
application.jar

Programming Tasks

Programming Task No 1

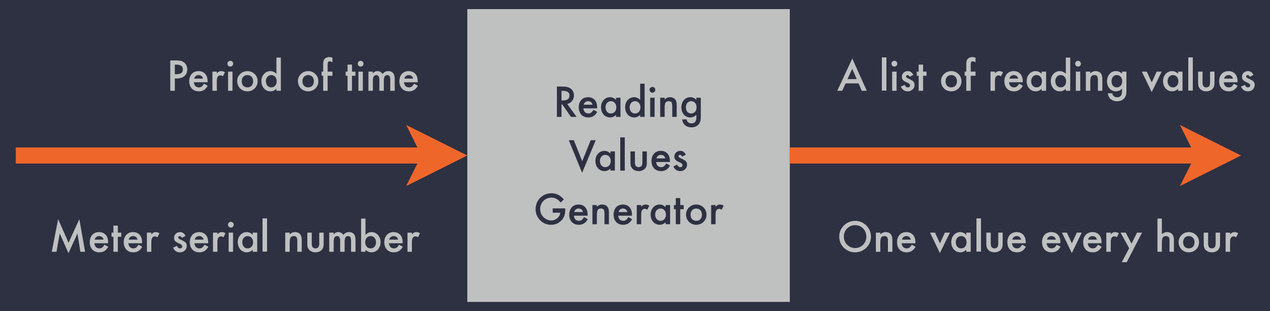
Programming Task No 1

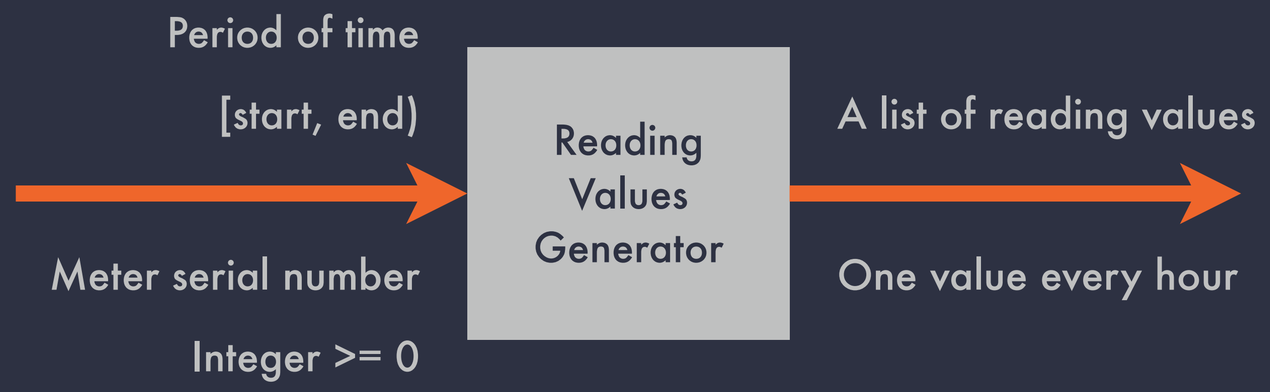
Generator Of Meter Reading Values

Reading Value

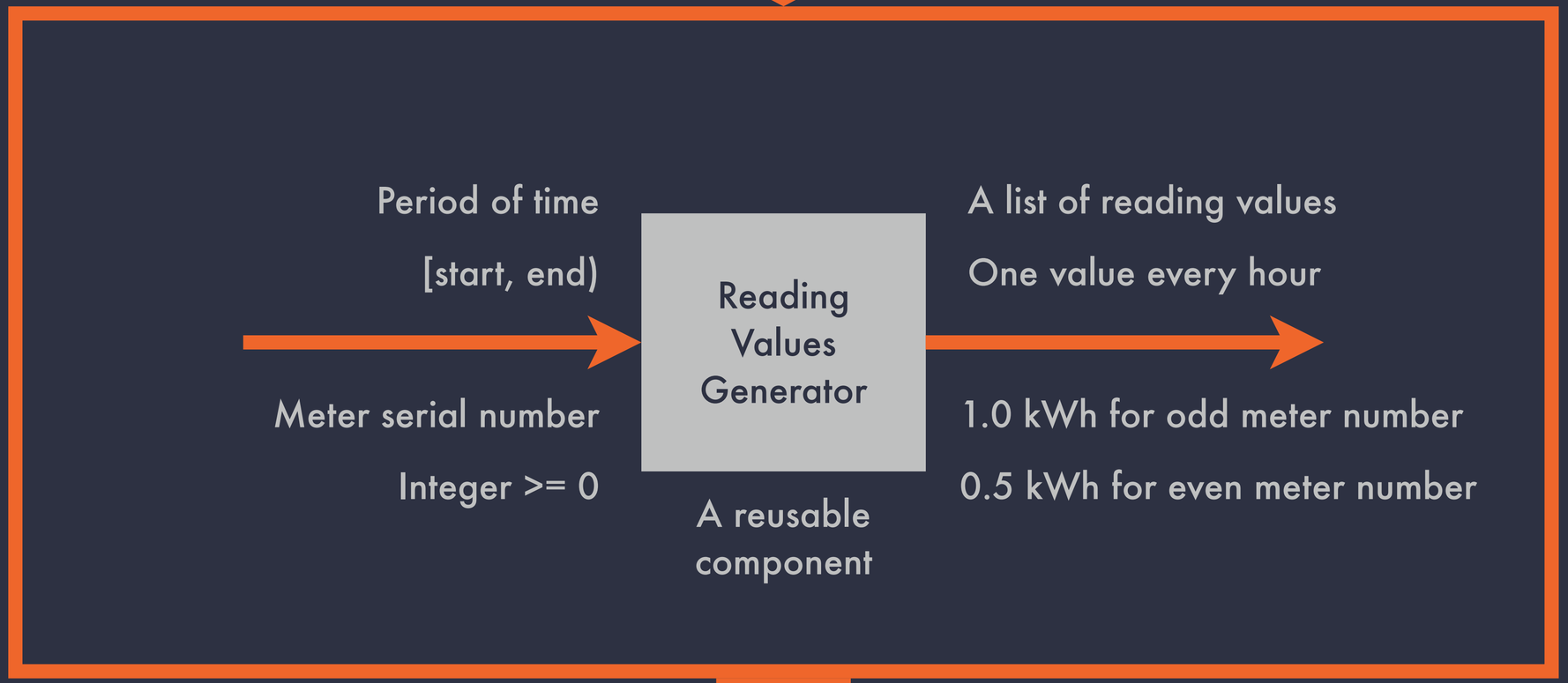
Value [kWh]

Timestamp


Period of time
Meter serial number
Reading Values Generator
A list of reading values
One value every hour


Period of time
[start, end)
Meter serial number
Integer >= 0
Reading Values Generator
A list of reading values
One value every hour

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

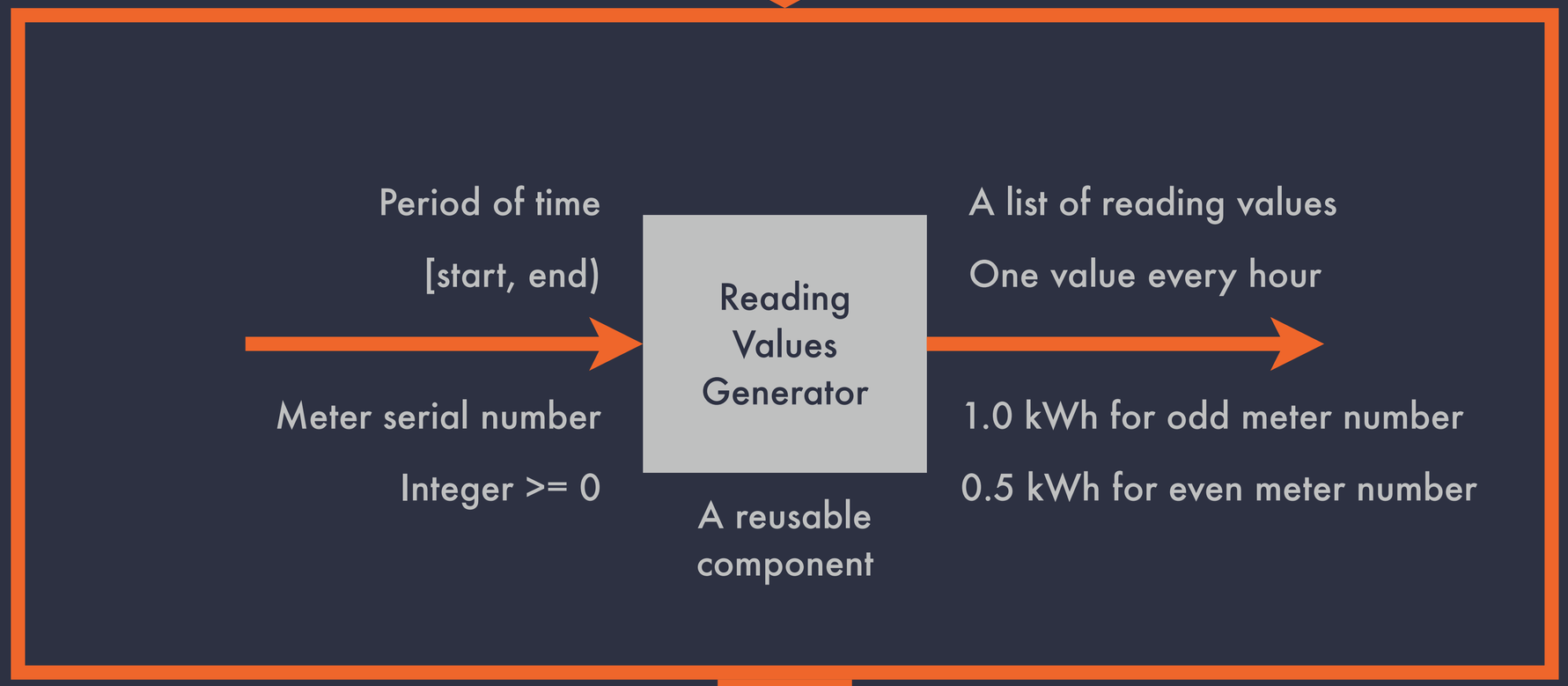


A console application

Print generated reading values on the screen

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

Meter serial number = 4512



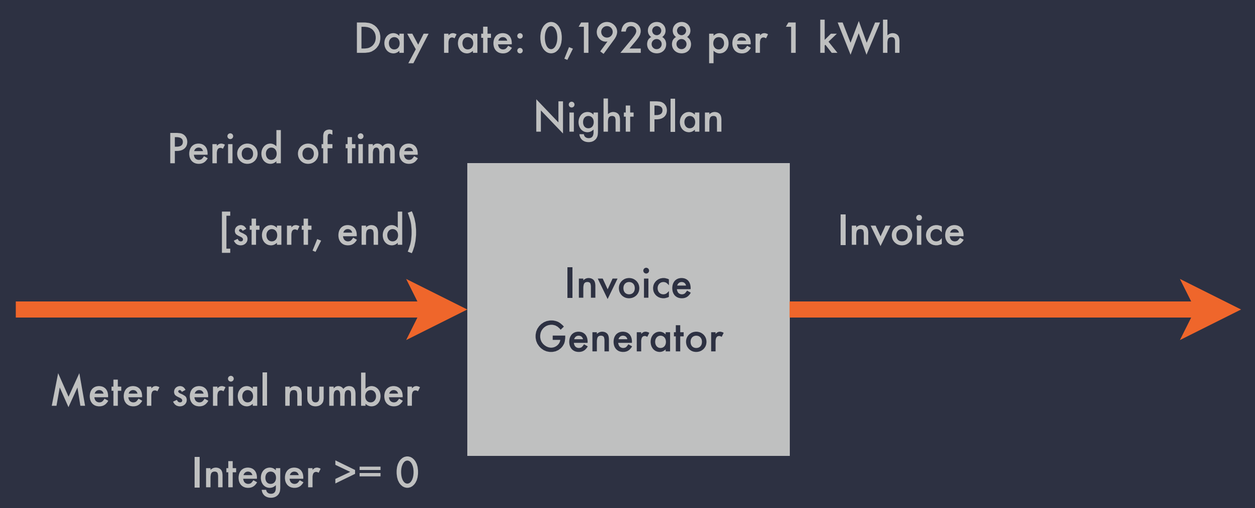
A console application

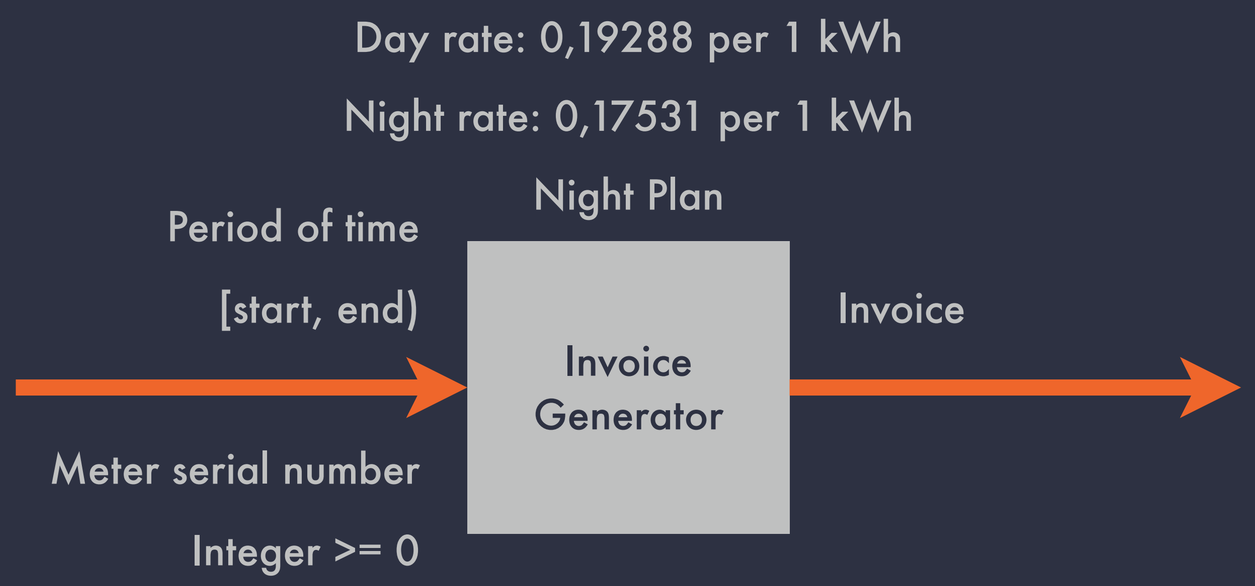
Print generated reading values on the screen

Programming Task No 2

Programming Task No 2

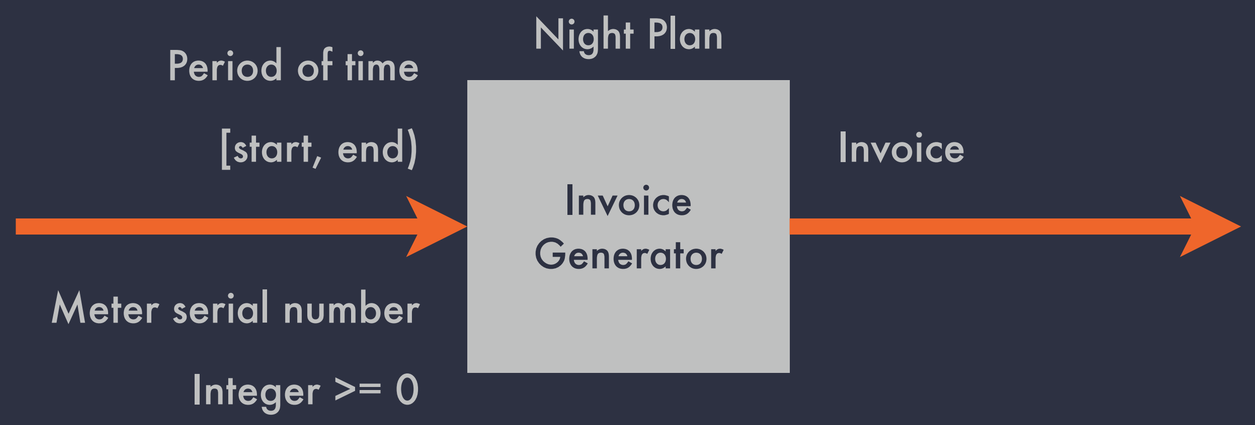
Invoice Generator


Day rate: 0,19288 per 1 kWh
Night Plan
Period of time
[start, end)
Meter serial number
Integer >= 0
Invoice Generator
Invoice


Day rate: 0,19288 per 1 kWh
Night rate: 0,17531 per 1 kWh
Night Plan
Period of time
[start, end)
Meter serial number
Integer >= 0
Invoice Generator
Invoice

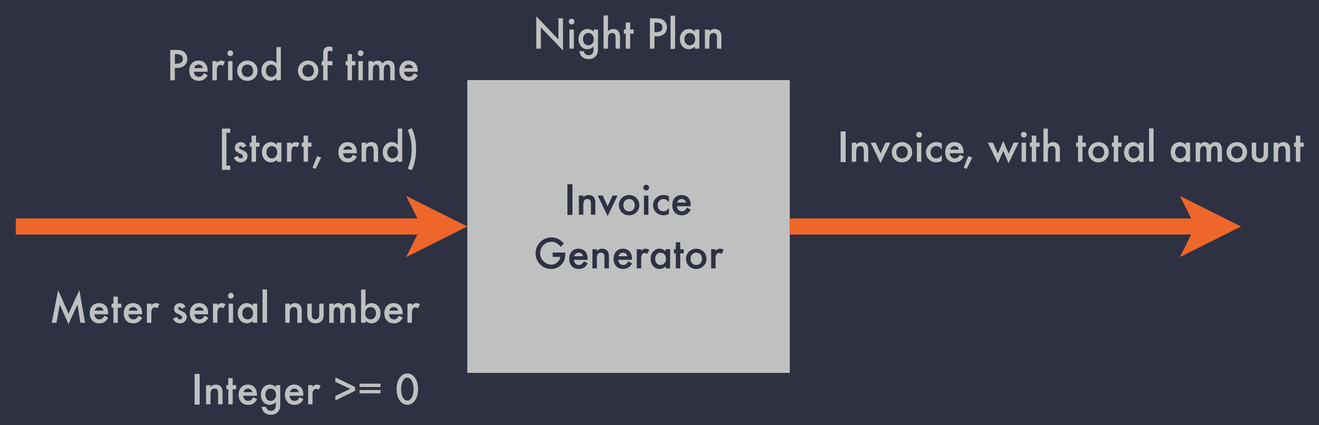
Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh



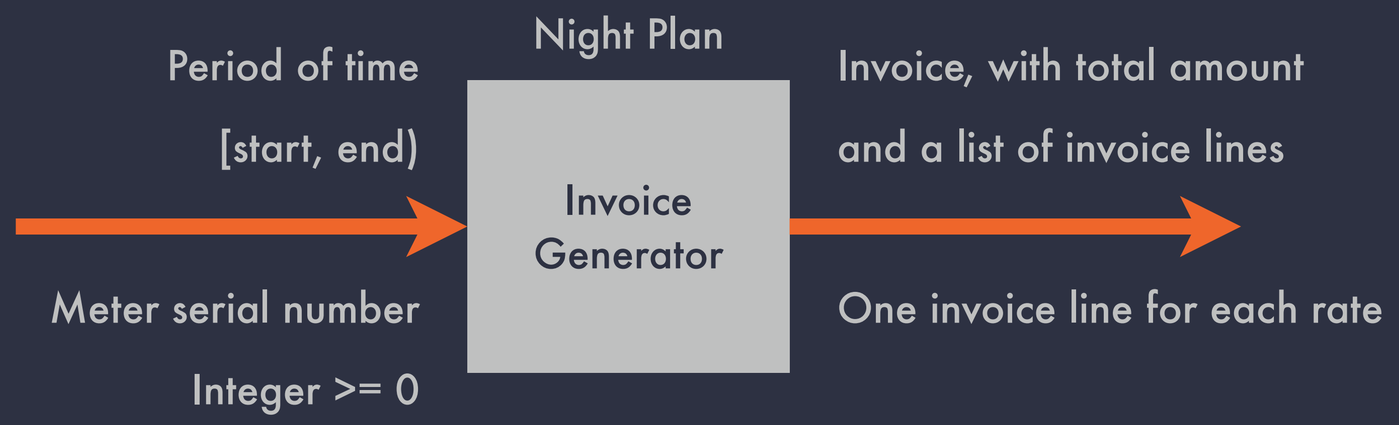
Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am



Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am



Invoice

Total amount

A list of invoice lines

Rate description in English
Rate value
Energy consumed [kWh]
Line amount

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

Meter serial number = 4512

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am

Night Plan

Period of time

[start, end)

Meter serial number

Integer >= 0

Invoice Generator

A reusable component

Invoice, with total amount

and a list of invoice lines

One invoice line for each rate

A console application

Print generated invoice with its invoice lines on the screen

* Make the design ready for another electricity consumption plan.

Unit Testing

Production code

Test code

Arrange
Act
Assert

Production code needs to be **testable**

Dependencies need to be **injectable**

ReadingValuesGenerator

From production code

ReadingValuesGeneratorStub

From test code

Test double

Dummy Stub Spy Mock Fake

```
public class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    public InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    ...
}
```

The technique is called: dependency injection

Debugger

When we don’t know the code so well…

When the code is too complex…

When the case is too complex…

Tips on Learning Programming

Technical foundations

Java language

Syntax, data types, collections

Java language

Syntax, data types, collections, threads

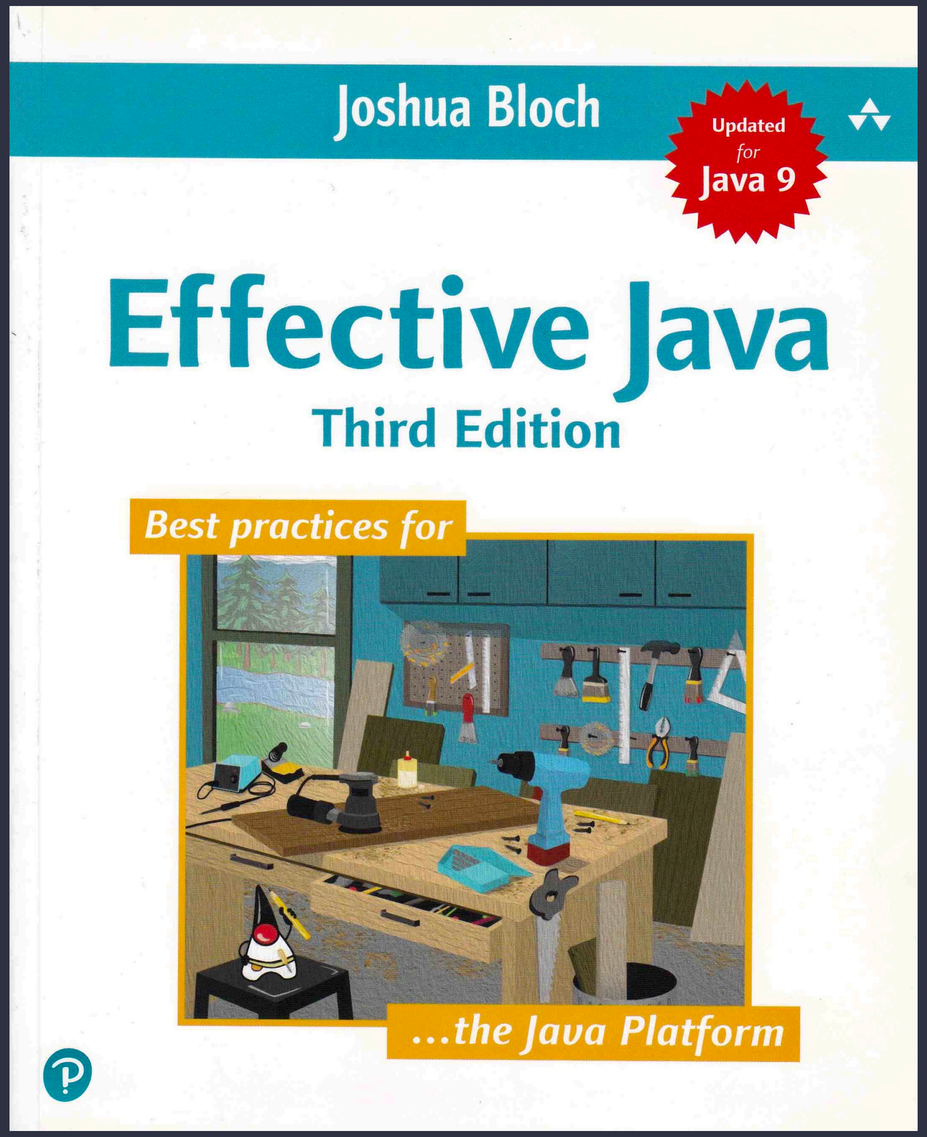
Practice a lot!

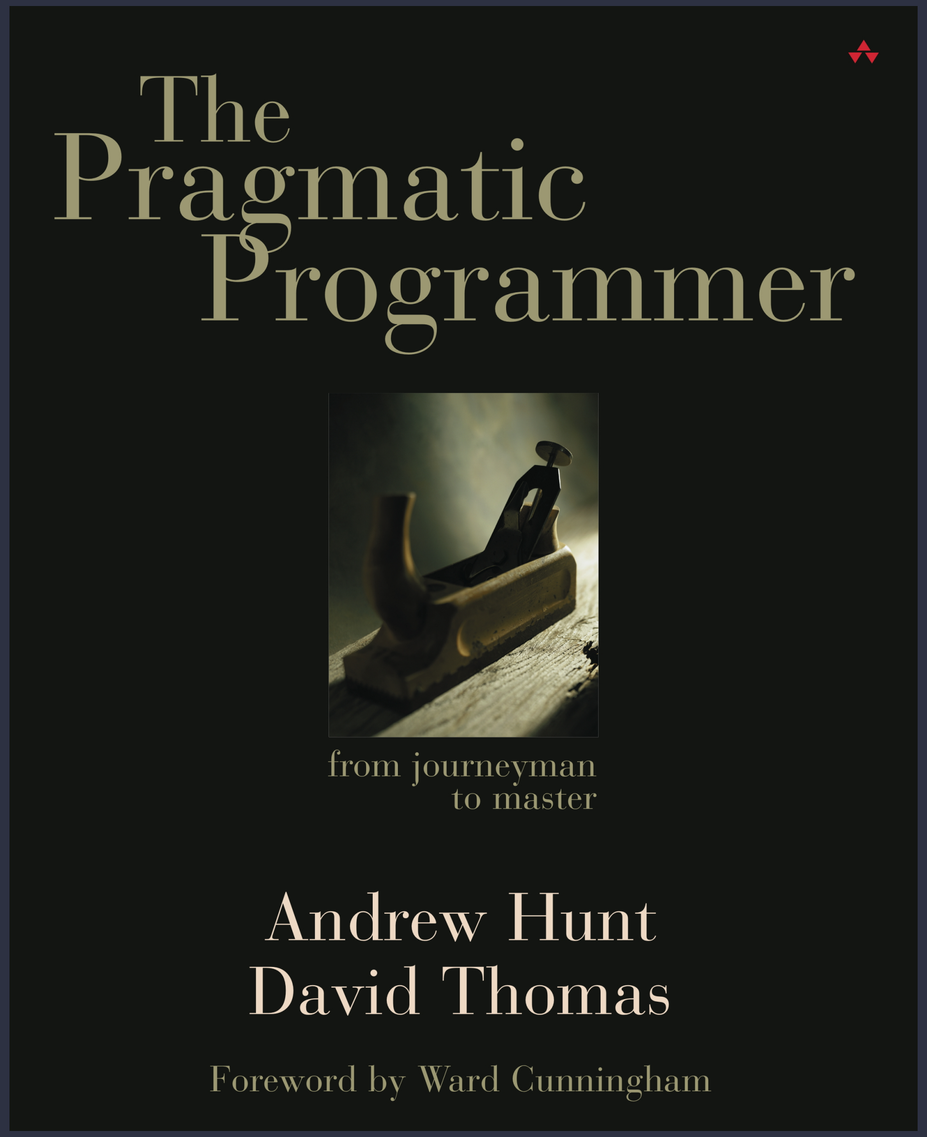
Algorithms

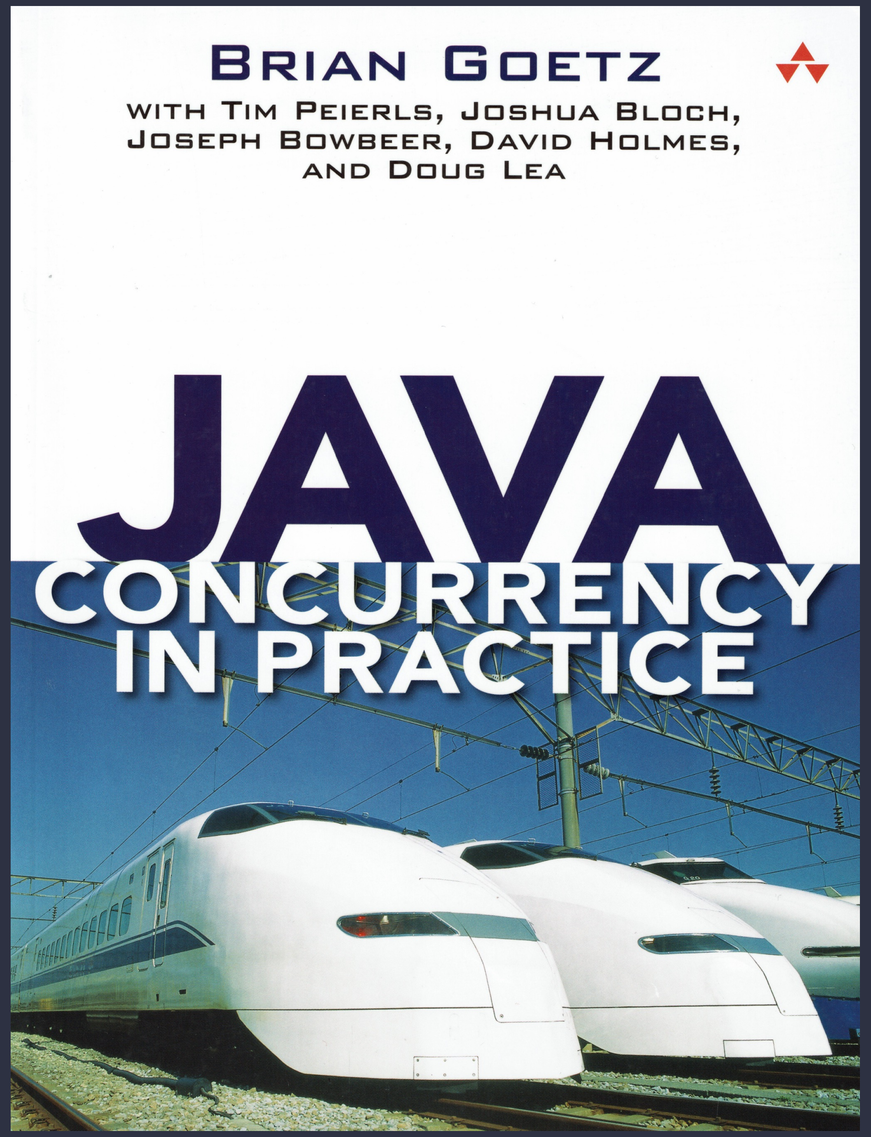
E.g. sorting algorithms, big O() notation

Watch out for nested loops!

Abstract thinking


Joshua Bloch
Updated for Java 9
Effective Java
Third Edition
Best practices for
...the Java Platform


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham


BRIAN GOETZ
WITH TIM PEIERLS, JOSHUA BLOCH,
JOSEPH BOWBEER, DAVID HOLMES,
AND DOUG LEA
JAVA
CONCURRENCY
IN PRACTICE

Practice, practice, practice!

Spring Framework

Many of enterprise applications share the same technical problems.

Solving technical problems

Solving problems of enterprises

There are many tools supporting our development processes and projects themselves.

Continuous Integration Servers
Artefacts Repositories
Version Control Systems

Frameworks
Application Servers
Messaging Systems
Databases
Monitoring Systems

Spring Framework

Fundamental Techniques, RESTful Services, Programming Task, Spring Project Review, Tips on Learning Spring

Fundamental Techniques: DI and IoC

DI

Dependency Injection

Auto-wiring

```java
public class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    public InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    ...
}
```

+ Setter injection

+ Field injection (Spring)

```
public class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    public InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    ...
}
```

Constructor injection

+ Setter injection

+ Field injection (Spring)

Where do the injected objects come from?

```
public class Application {

    public static void main(String[] args) {

        (...)

        ReadingValuesGenerator readingValuesGenerator = new ReadingValuesGenerator();
        InvoiceGenerator invoiceGenerator = new InvoiceGenerator(readingValuesGenerator);
        Invoice invoice = invoiceGenerator.generateInvoice(...);
        System.out.println("invoice = " + invoice);
    }
}
```

```
public class Application {

    public static void main(String[] args) {

        (...)

        ReadingValuesGenerator readingValuesGenerator = new ReadingValuesGenerator();
        InvoiceGenerator invoiceGenerator = new InvoiceGenerator(readingValuesGenerator);
        Invoice invoice = invoiceGenerator.generateInvoice(...);
        System.out.println("invoice = " + invoice);
    }
}
```

Manual flow control

1) explicitly lay out all of the steps in the source code

2) manually create ReadingValuesGenerator

3) manually create InvoiceGenerator passing ReadingValuesGenerator

Manual flow control

1) explicitly lay out all of the steps in the source code

2) manually create ReadingValuesGenerator

3) manually create InvoiceGenerator passing ReadingValuesGenerator

Inversion of control

1) discover InvoiceGenerator and its dependency to ReadingValuesGenerator

2) automatically create ReadingValuesGenerator

3) automatically create InvoiceGenerator injecting ReadingValuesGenerator

Inversion of control

1) discover InvoiceGenerator and its dependency to ReadingValuesGenerator

2) automatically create ReadingValuesGenerator

3) automatically create InvoiceGenerator injecting ReadingValuesGenerator

```
@Component
public class ReadingValuesGenerator {

    (...)

}

@Component
public class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;
    (...)
    @Autowired
}   public InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    (...)

}
```

REST

Tim Berners-Lee, 1989

HTML

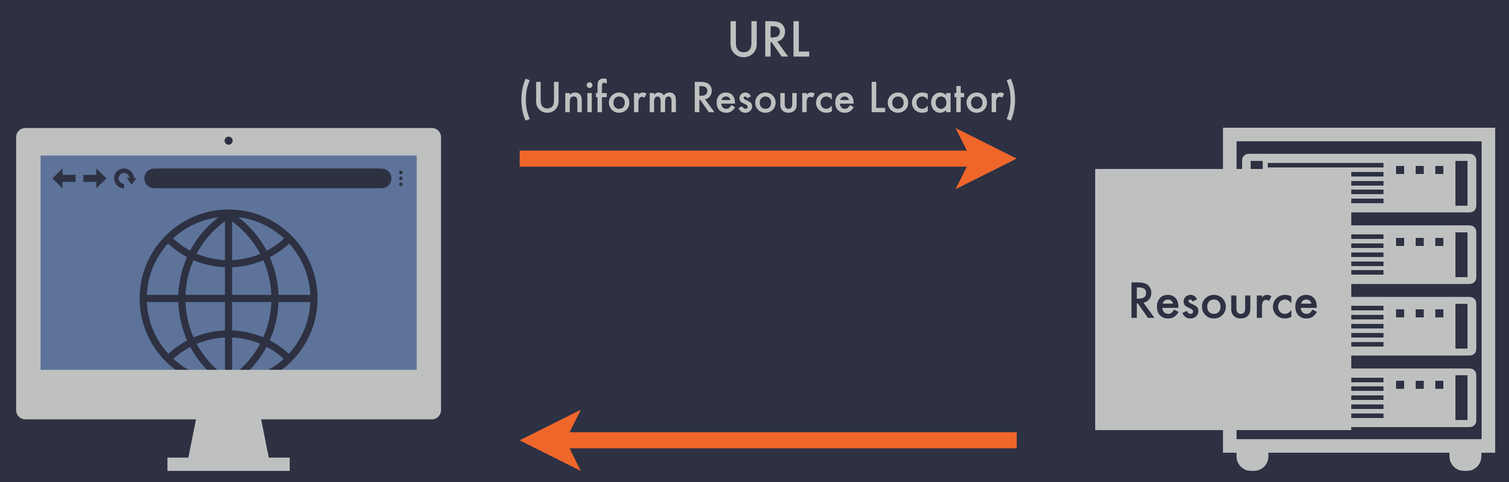
HyperText Markup Language

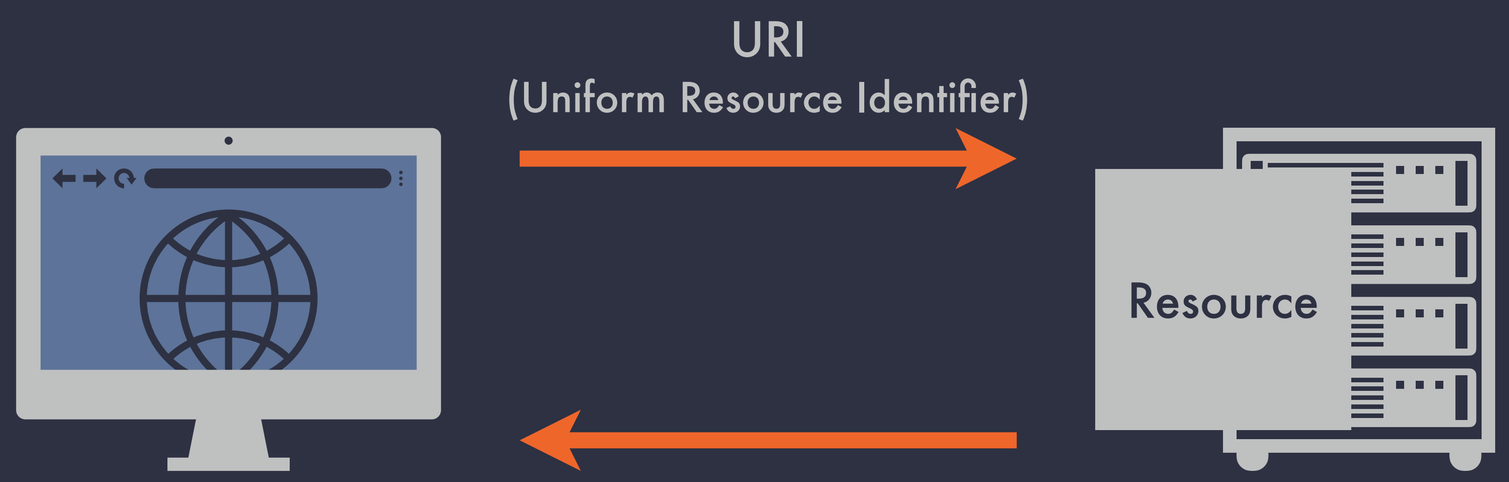
HTTP

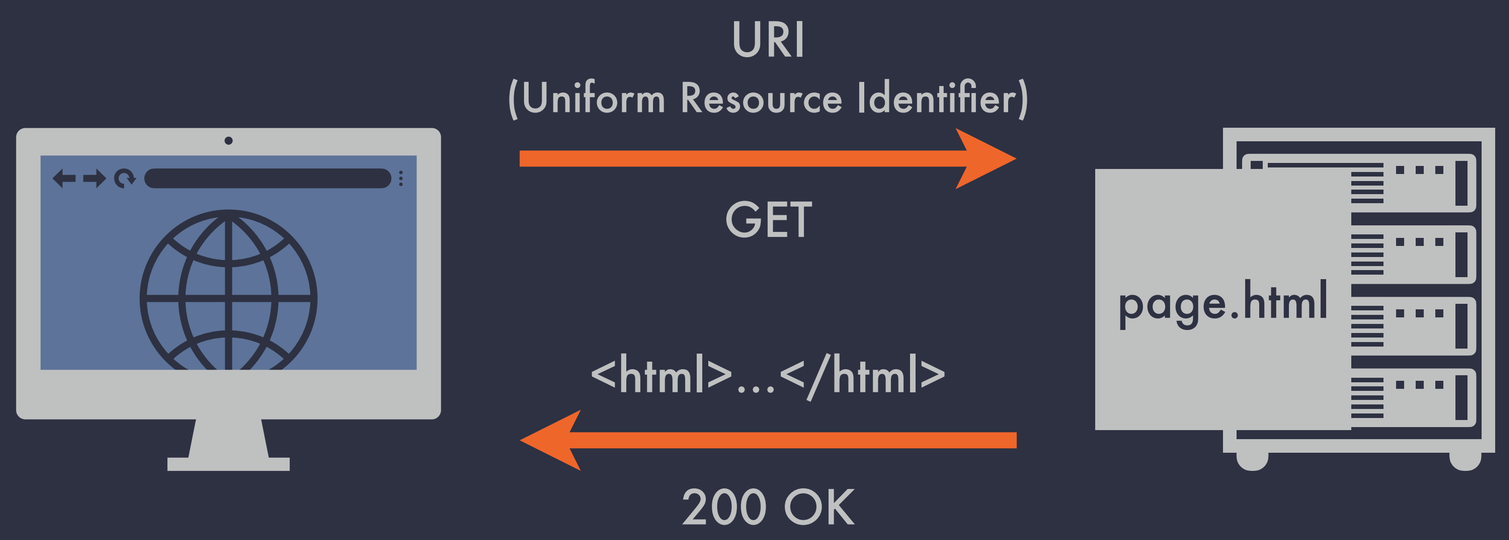
HyperText Transfer Protocol

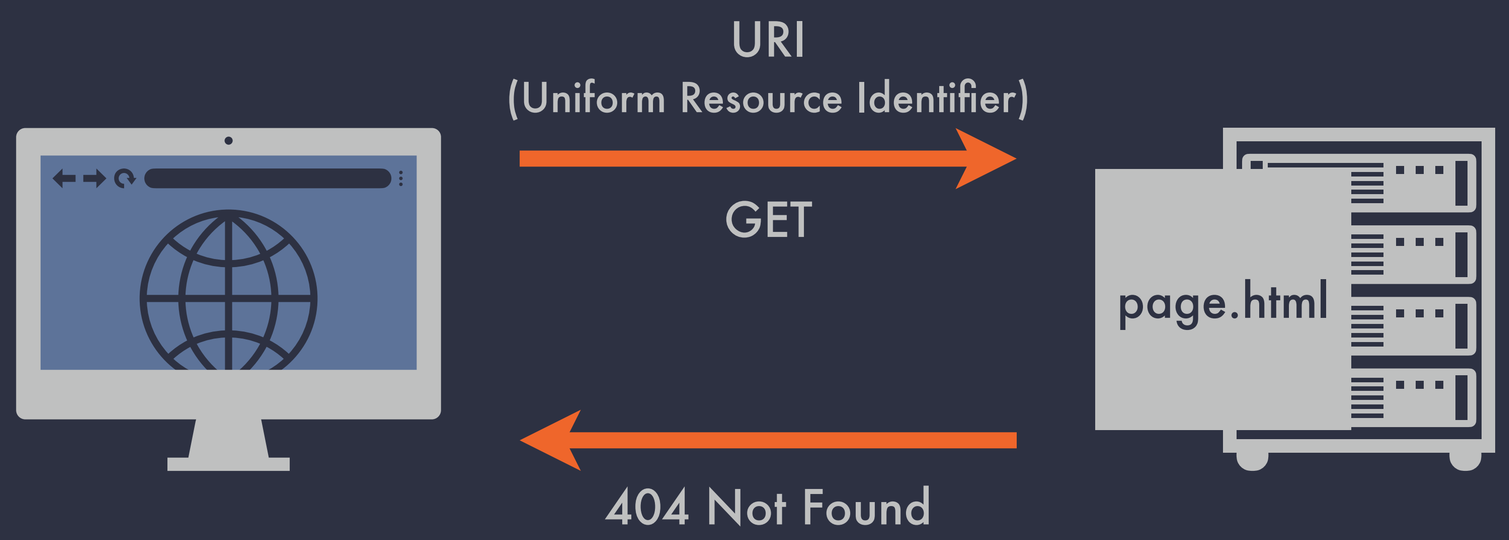
Address

Address
Resource

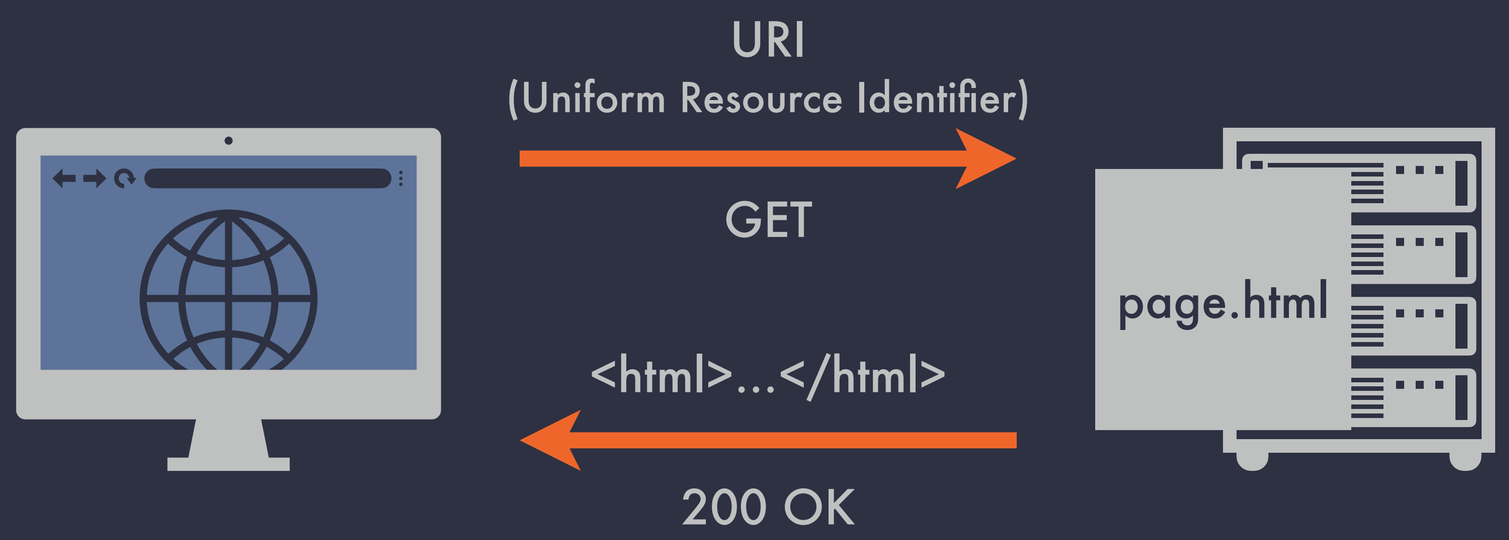

URL
(Uniform Resource Locator)
Resource

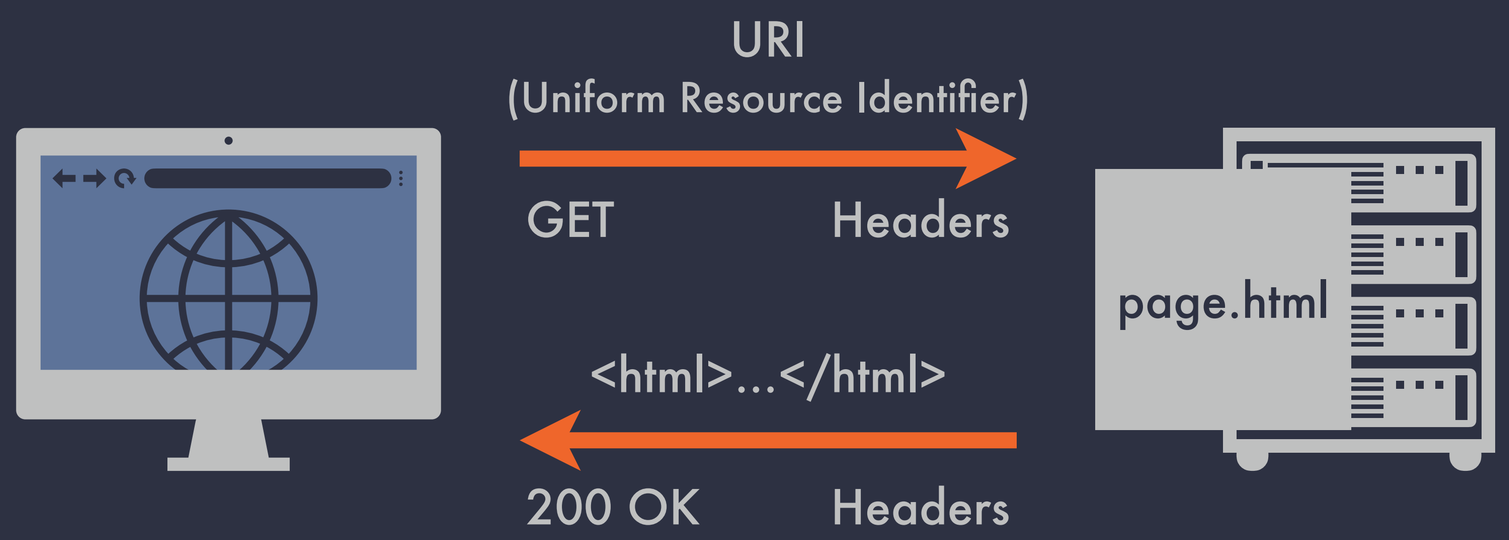

URI
(Uniform Resource Identifier)
Resource

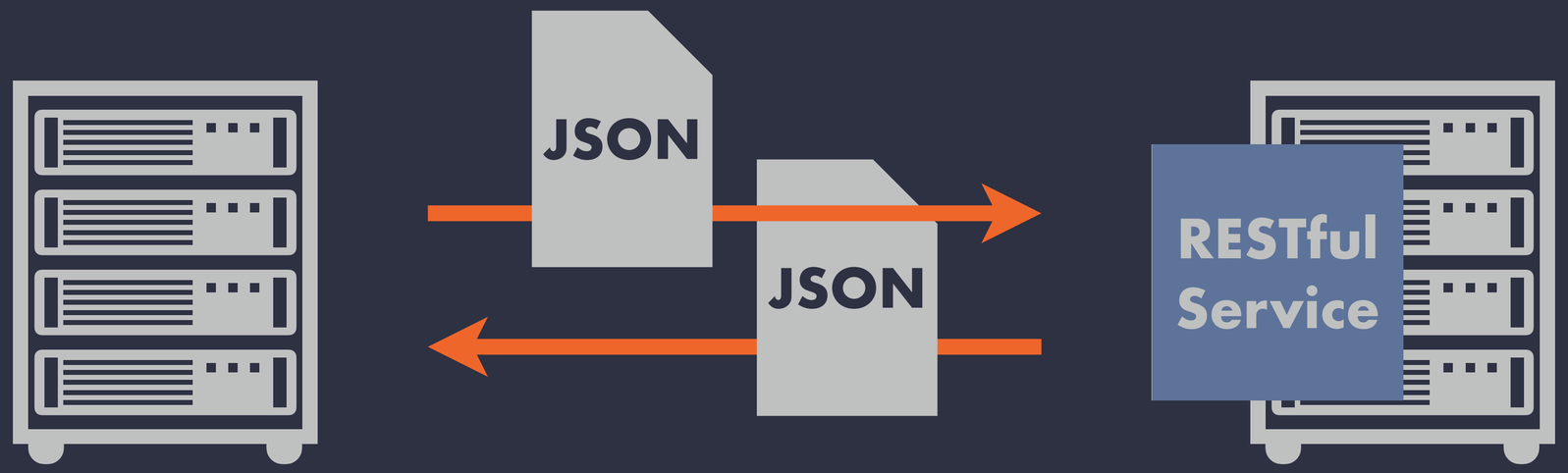

URI
(Uniform Resource Identifier)
GET
page.html
<html>...</html>
200 OK


URI
(Uniform Resource Identifier)
GET
page.html
404 Not Found

1xx	
2xx	Success
3xx	
4xx	Client error
5xx	Server error


URI
(Uniform Resource Identifier)
GET
page.html
<html>...</html>
200 OK


URI
(Uniform Resource Identifier)
GET
Headers
page.html
<html>...</html>
200 OK
Headers

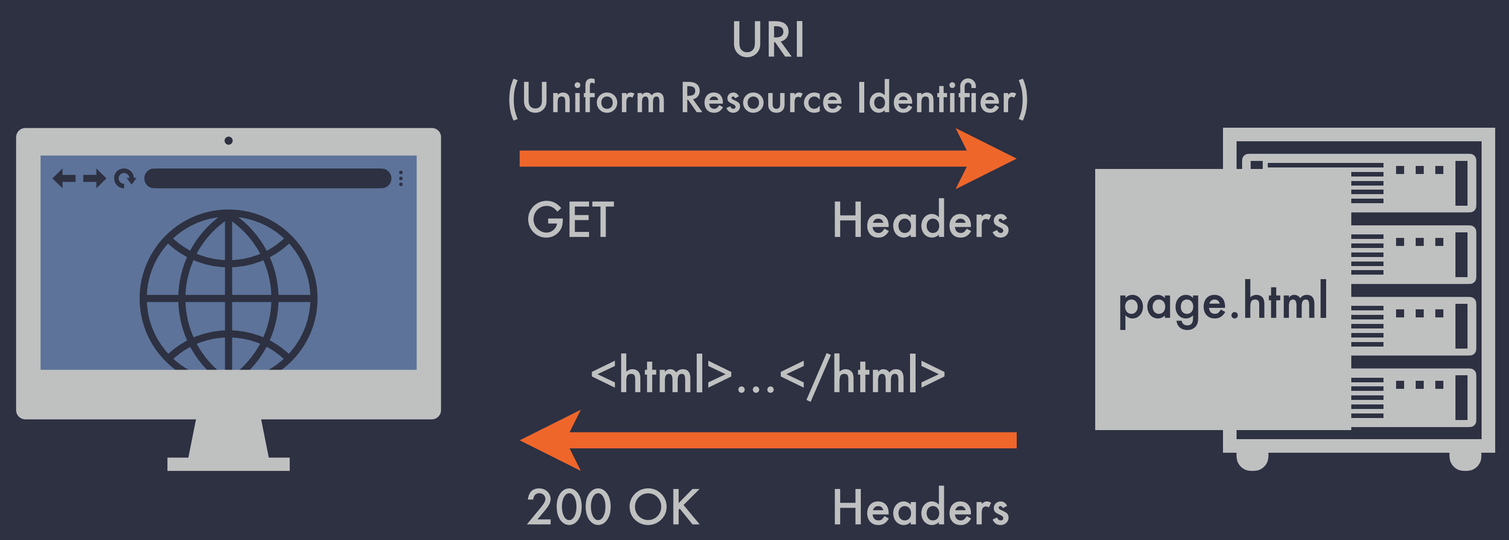

JSON
JSON
RESTful
Service

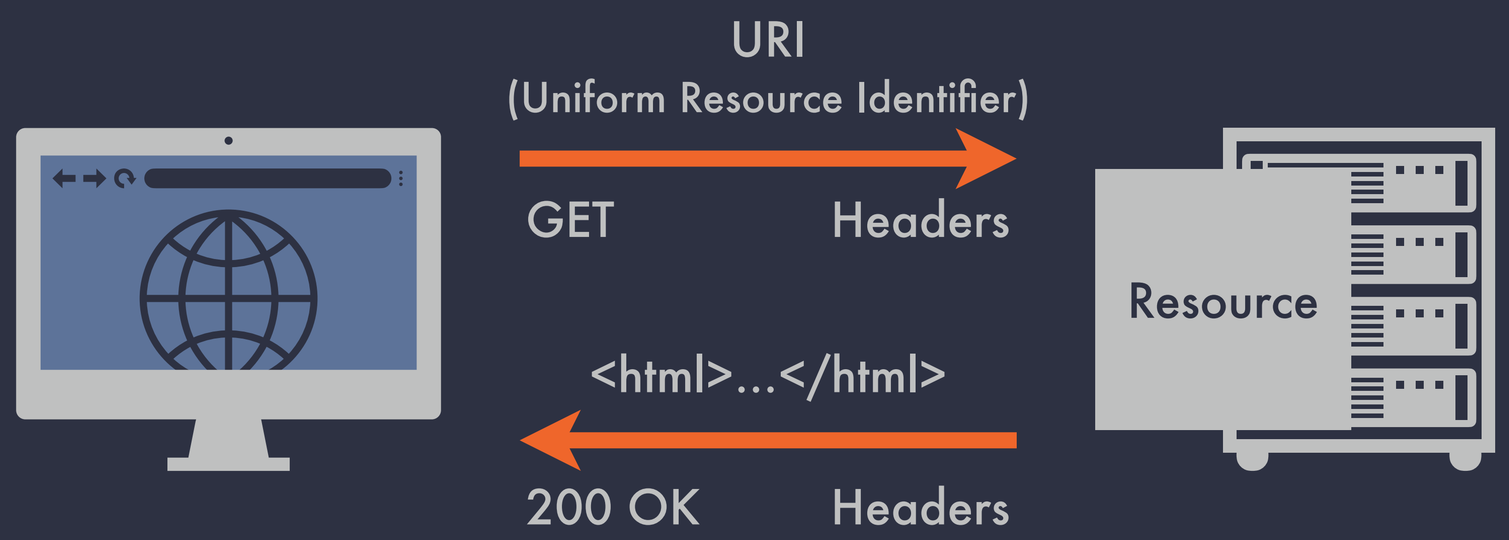
REST

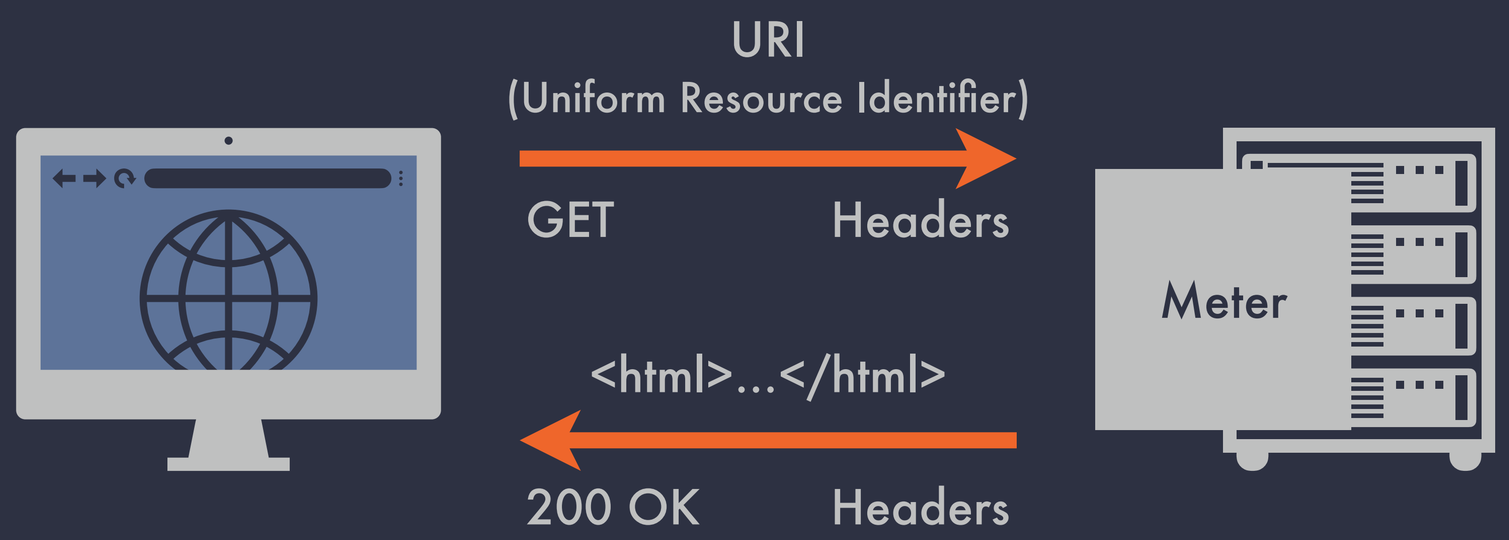
REST

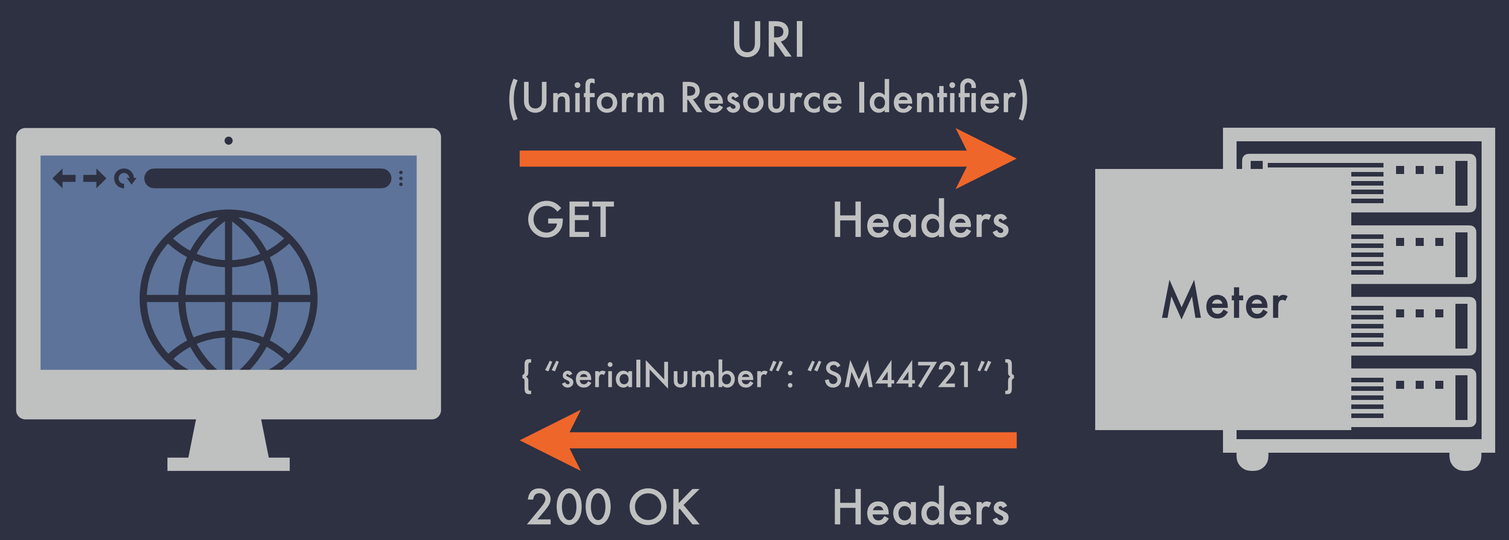
REpresentational State Transfer

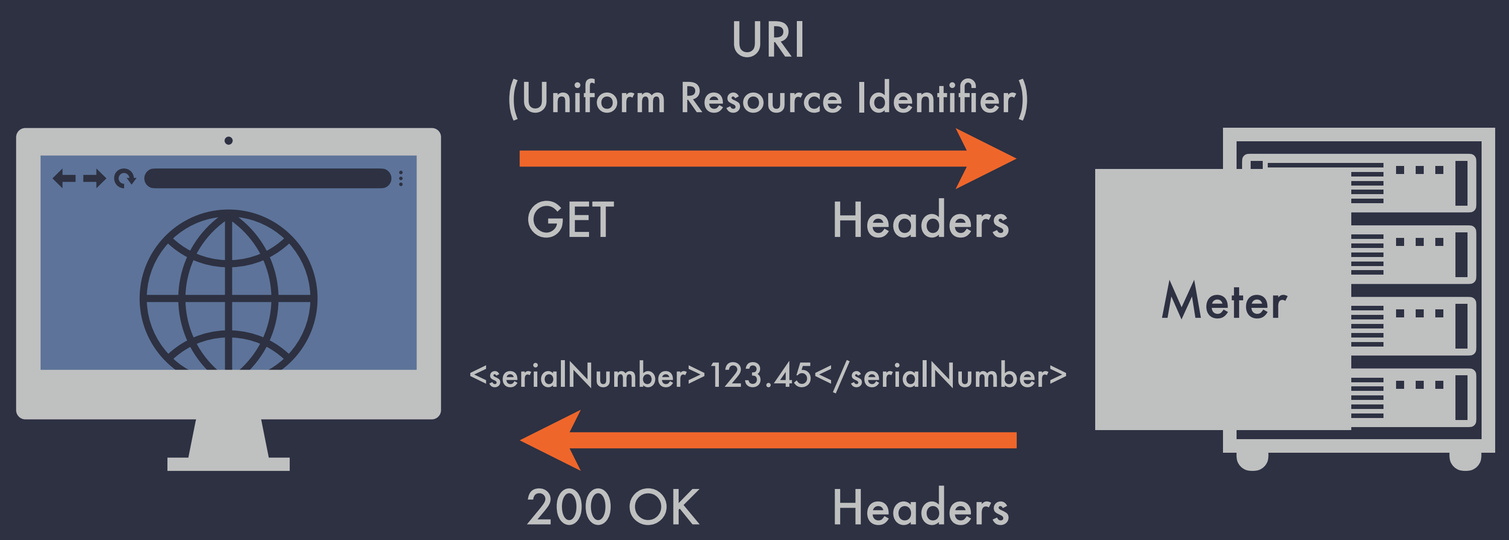
Roy Fielding, 2000


URI
(Uniform Resource Identifier)
GET
Headers
page.html
<html>...</html>
200 OK
Headers


URI
(Uniform Resource Identifier)
GET
Headers
Resource
<html>...</html>
200 OK
Headers


URI
(Uniform Resource Identifier)
GET
Headers
Meter
<html>...</html>
200 OK
Headers


URI
(Uniform Resource Identifier)
GET
Headers
Meter
{ “serialNumber”: “SM44721” }
200 OK
Headers


URI
(Uniform Resource Identifier)
GET
Headers
Meter
<serialNumber>123.45</serialNumber>
200 OK
Headers

POST	CREATE
GET	READ
PUT	UPDATE
DELETE	DELETE

POST

{ “id”: 42583 }

201 Created

POST

200 OK

/api/invoiceRuns

Application Programming Interface

/api/invoiceRuns/42583/start

Programming Task

Programming Task

Invoices in Spring

Create a new Spring Boot application.

Copy code from previous projects.

Create 3 invoices.

Expose invoices at /api/invoices via a GET method.

Create invoices during application startup.

Make invoices available in a fake database repository.

Convert reading values and invoice generators to Spring components.

Clean up and repackage the code.

Spring in NrgyInvoicR

Tips on Learning Spring

Feel comfortable in Java

Spring modules and features

Practice a lot!

stackoverflow.com

stackoverflow.com

Official Spring documentation

Guides

Complete documentation

baeldung.com

vladmihalcea.com

Source code for NrgyInvoicR

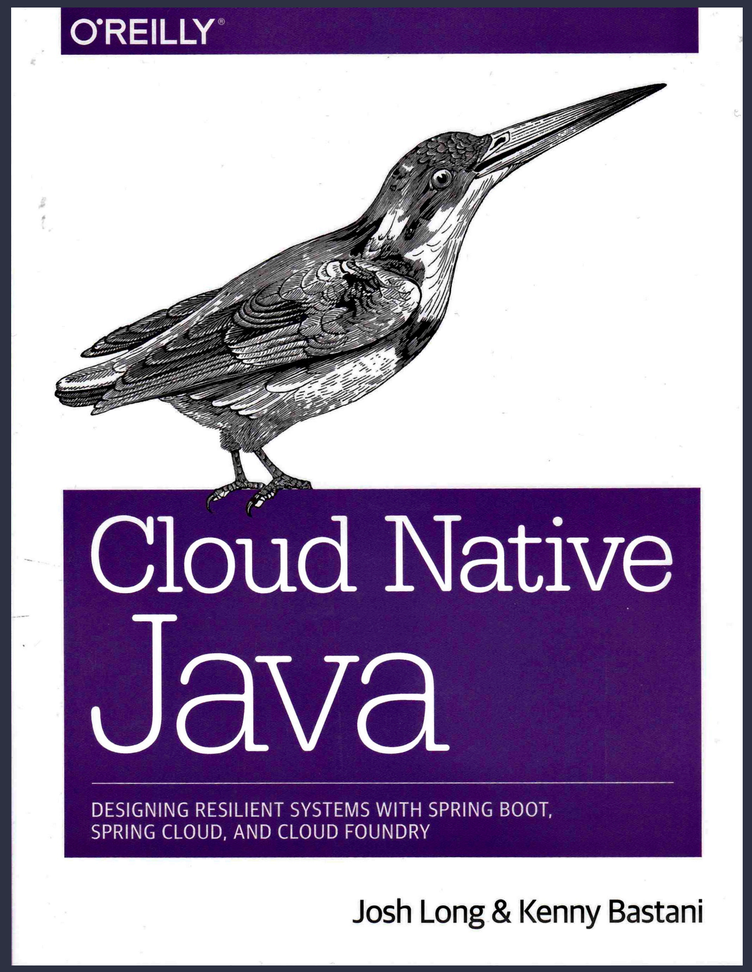
Read code

Josh Long

https://youtu.be/5q8B6lYhFvE

Josh Long

https://youtu.be/5q8B6lYhFvE



Frontend Development

Languages Overview

Languages Overview

HTML

HyperText Markup Language

Tags

CSS

Cascading Style Sheets

Styles HTML

Layout

Flexbox Layout Module

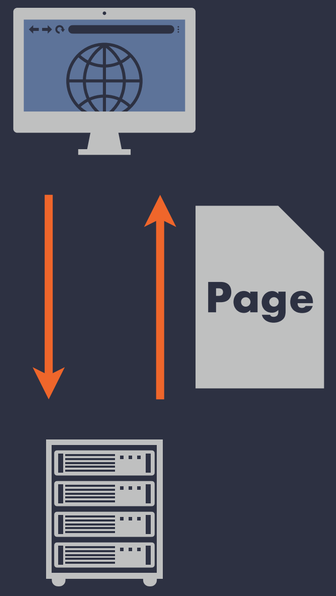
https://css-tricks.com/snippets/css/a-guide-to-flexbox

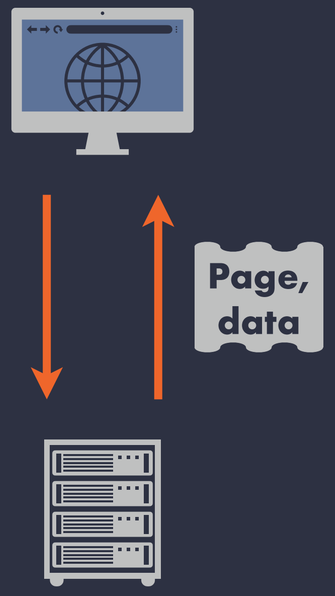
JavaScript

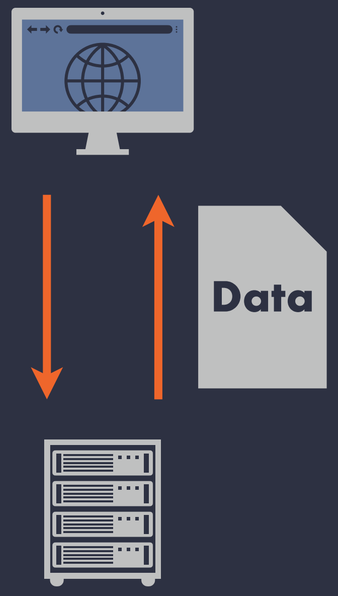
Programming Language for the Web

Controls HTML and CSS

~~Java~~







jQuery

2006

React.js

Library

2013

2000

2005

2010

2014

Frameworks

JSP

JSF

Struts

Spring MVC

AJAX

Asynchronous JavaScript and XML

AngularJS ≠ Angular 2

SPA

Single Page Applications

Breaking changes

TypeScript

TypeScript

TypeScript

Programming language designed by Microsoft

Superset of JavaScript

Statically typed

Object-oriented

(encapsulation, inheritance, polymorphism)

Angular Material

Angular Material

UI Components

https://material.angular.io

Data table with pagination

https://blog.angular-university.io/angular-material-data-table/

Programming Task

Programming Task

Invoices in Angular

Create a new Angular web application.

Display a menu with two options: Dashboard and Invoices.

Create two components: Dashboard component and Invoices component.

Fetch invoices from Spring application API during Invoices component initialisation.

For each invoice, display its total amount and all of its lines with all their attributes.

Use Angular Development Server.

Embed Angular web application in Spring application.

Tips on Learning Frontend Development

HTML, CSS

HTML, CSS, JavaScript

Angular, TypeScript

JavaScript

Java

Angular, TypeScript

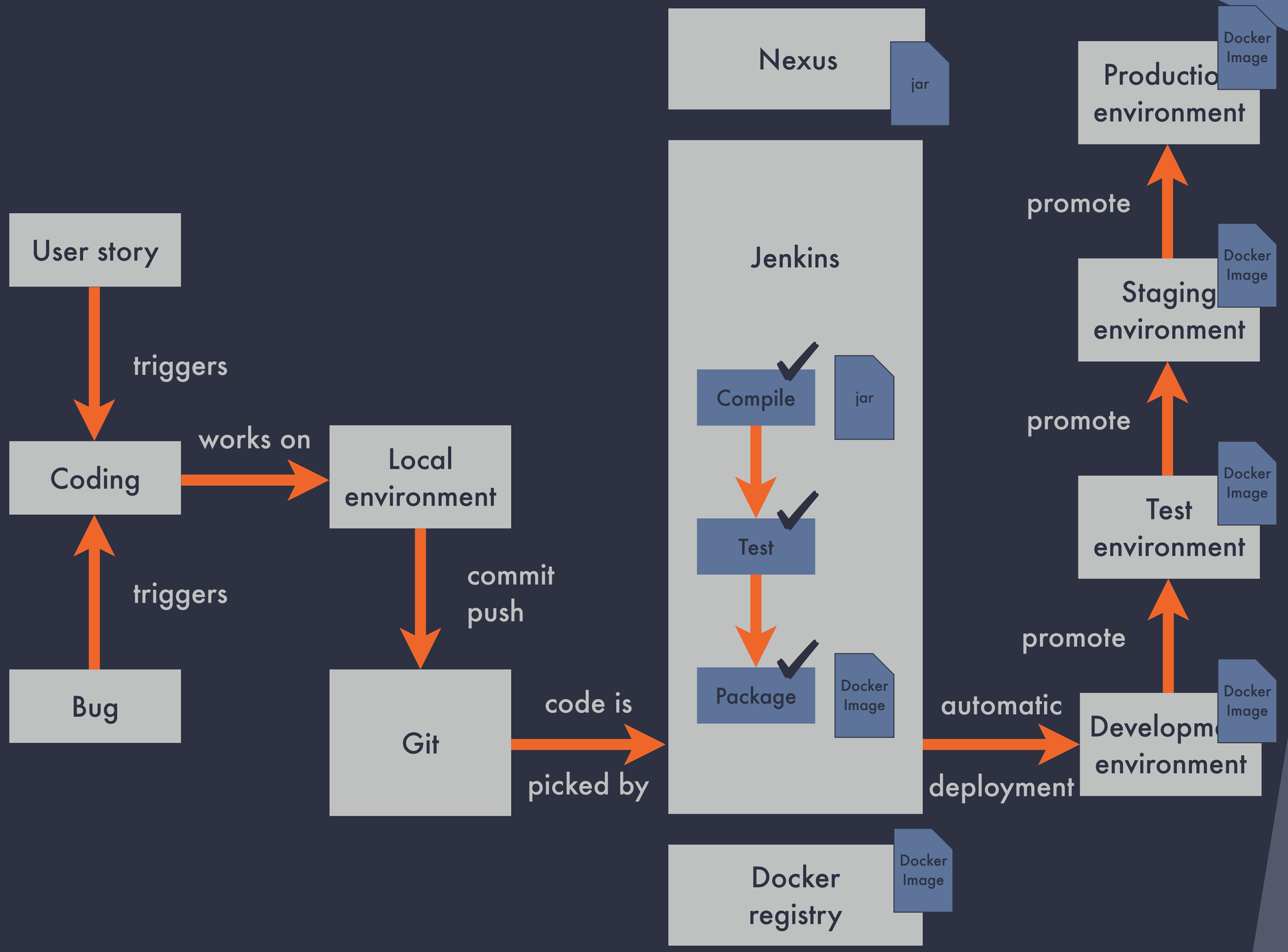
Practice a lot!

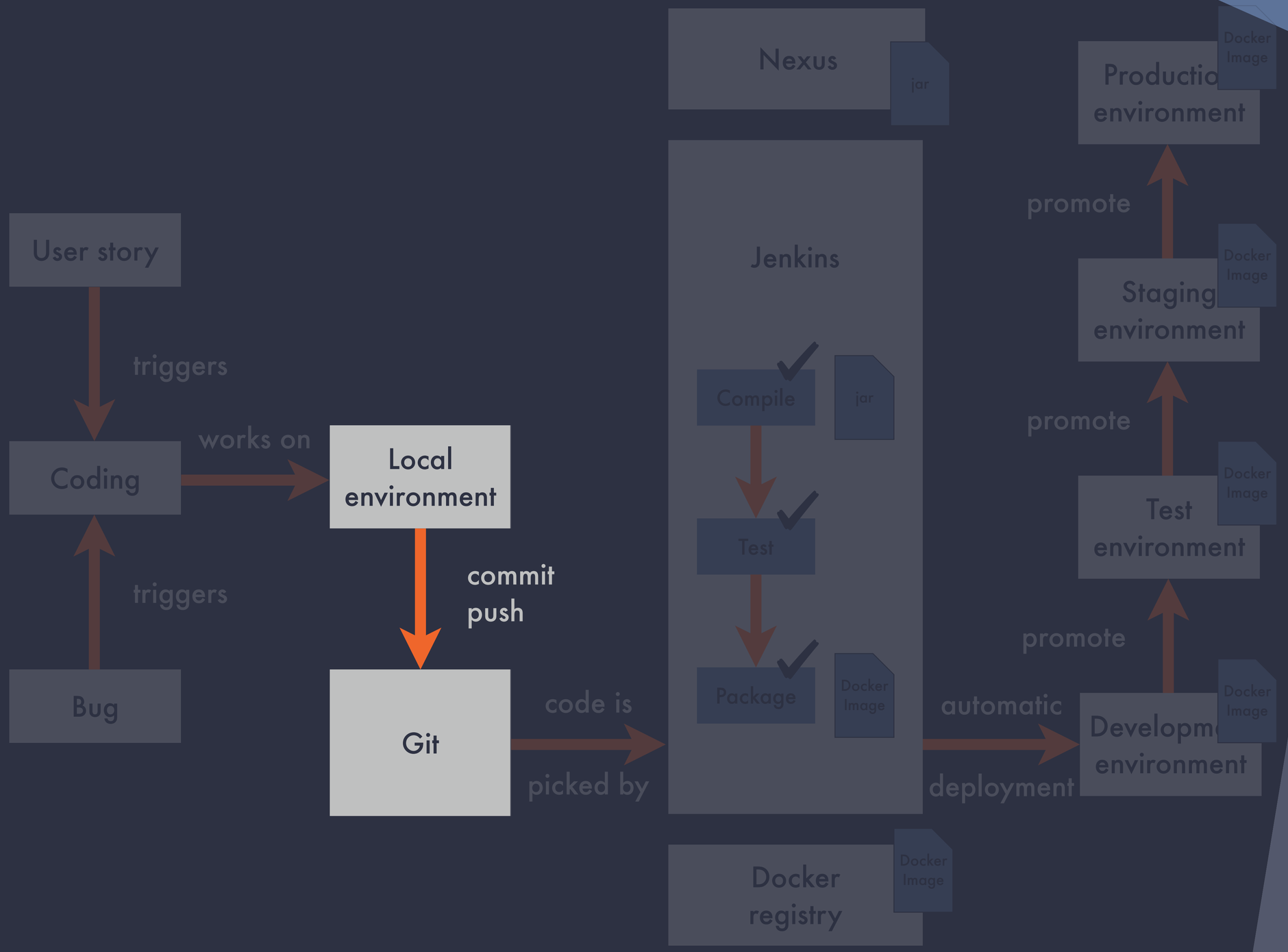
angular.io

material.angular.io

Source code for NrgyInvoicR

GIT


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image


User story
triggers
Coding
works on
Local environment
commit
push
triggers
Bug
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image

Version Control Systems

Version Control Systems

CVS

Concurrent Versions Systems

1990

SVN

Subversion

2004

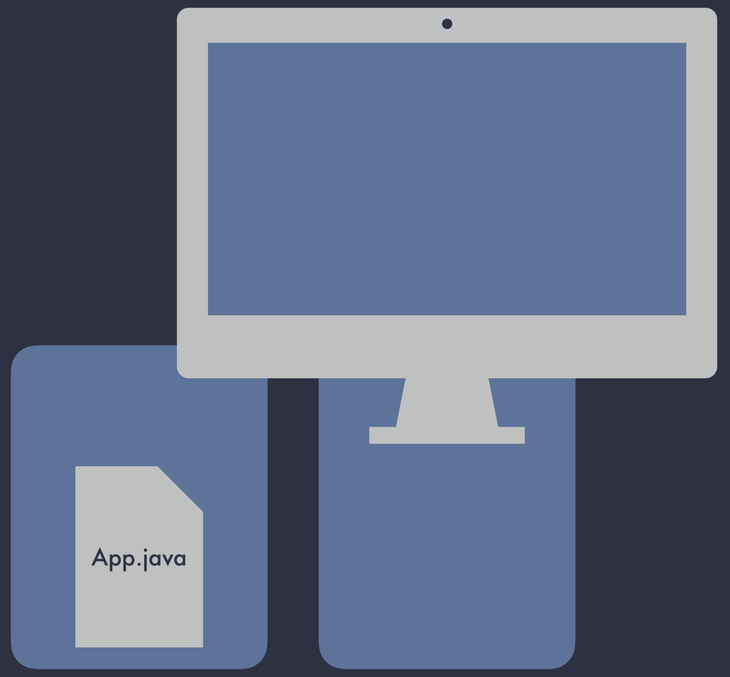
Require centralised server

Git

2005

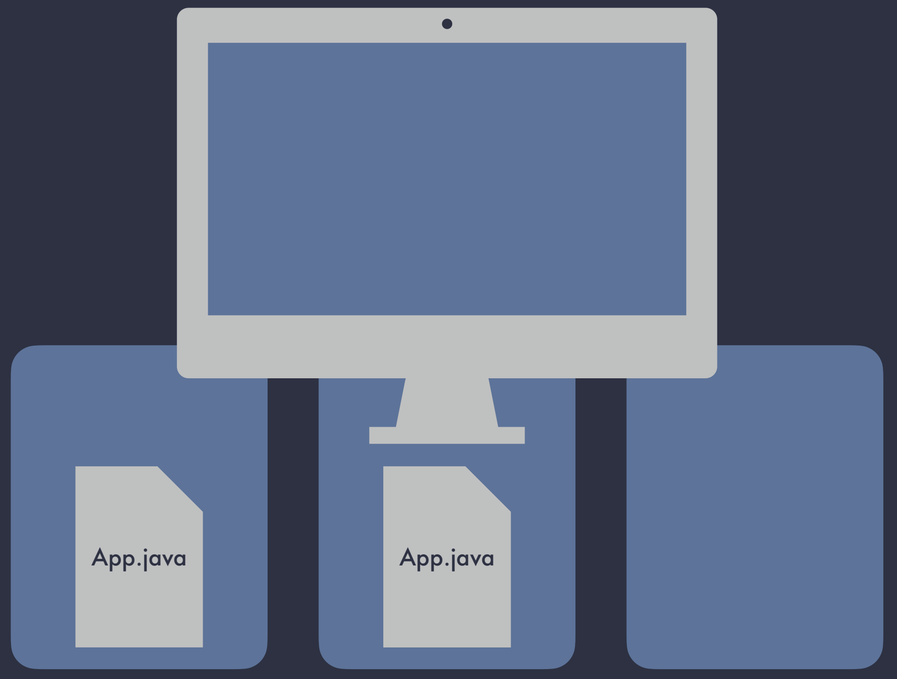
Distributed

Keeps entire history
of the project locally



Working directory
Working tree

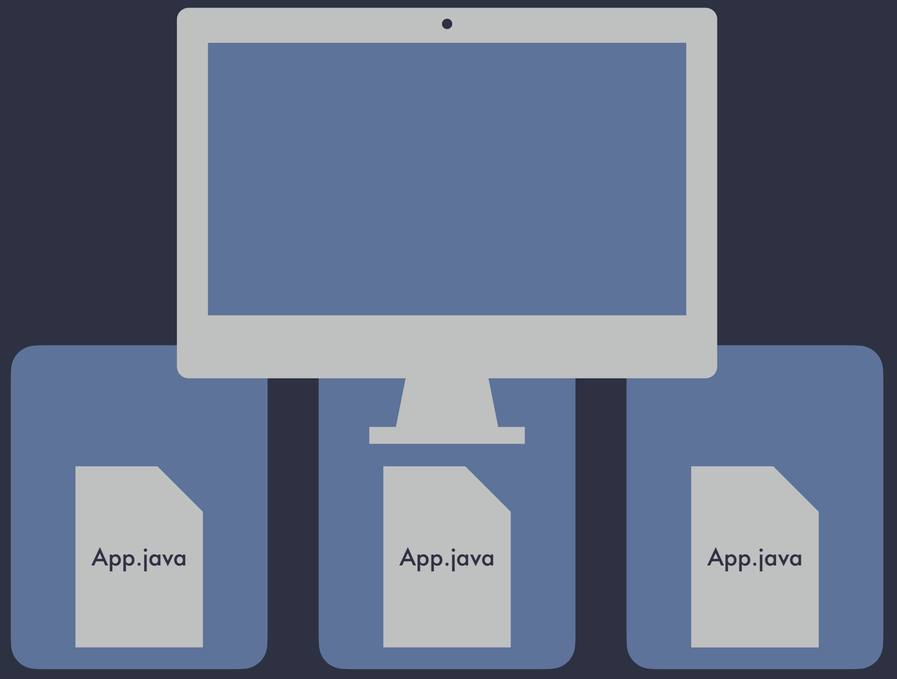
Index
Staging area



Working directory
Working tree

Index
Staging area

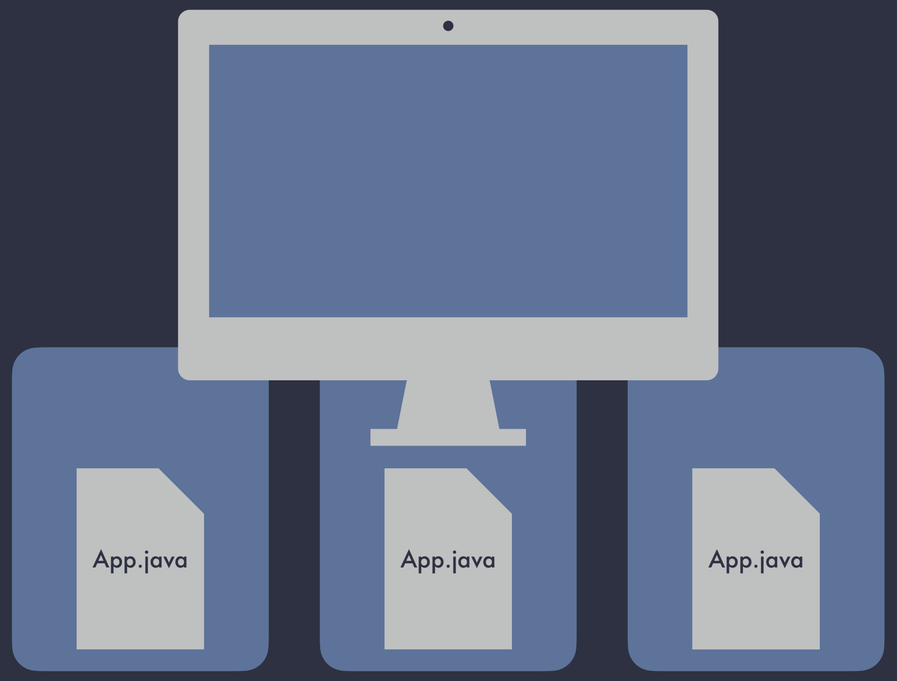
Local repository



Working directory
Working tree

Index
Staging area

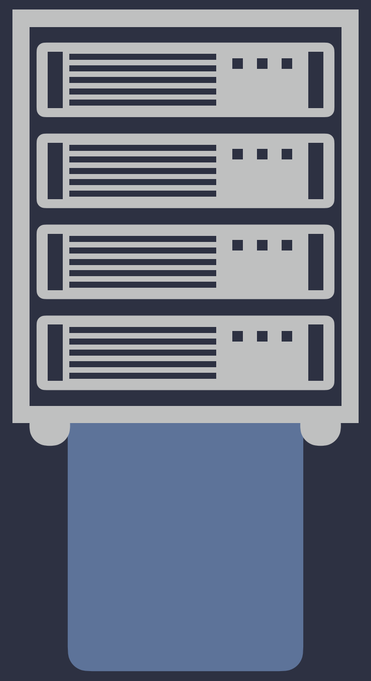
Local repository



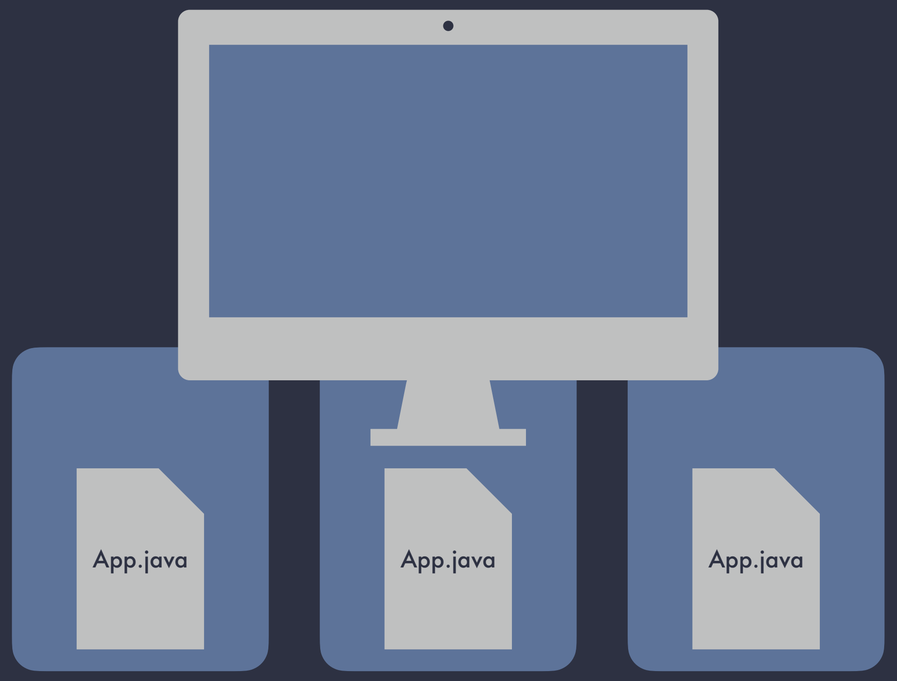
Working directory
Working tree

Index
Staging area

Local repository



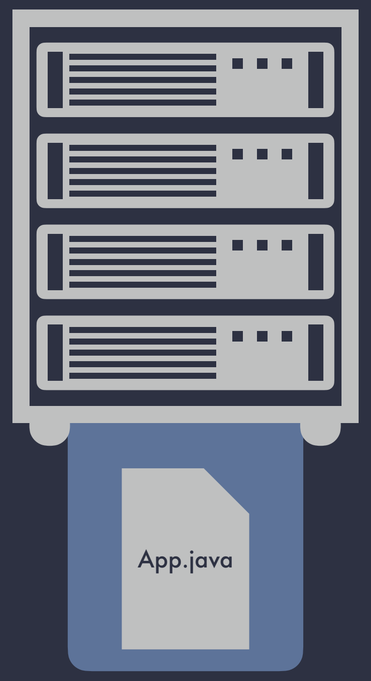
Remote repository



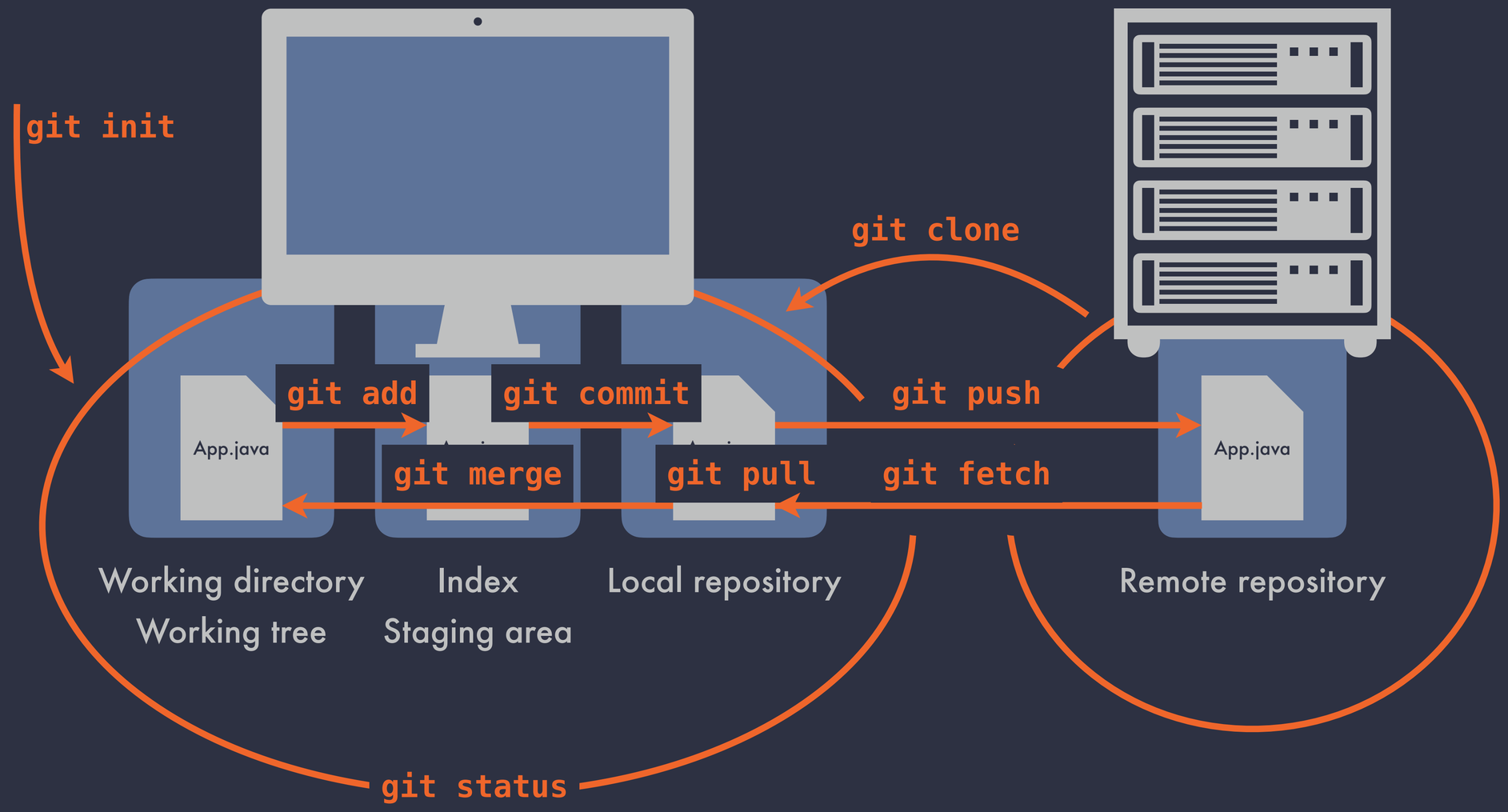
Working directory
Working tree

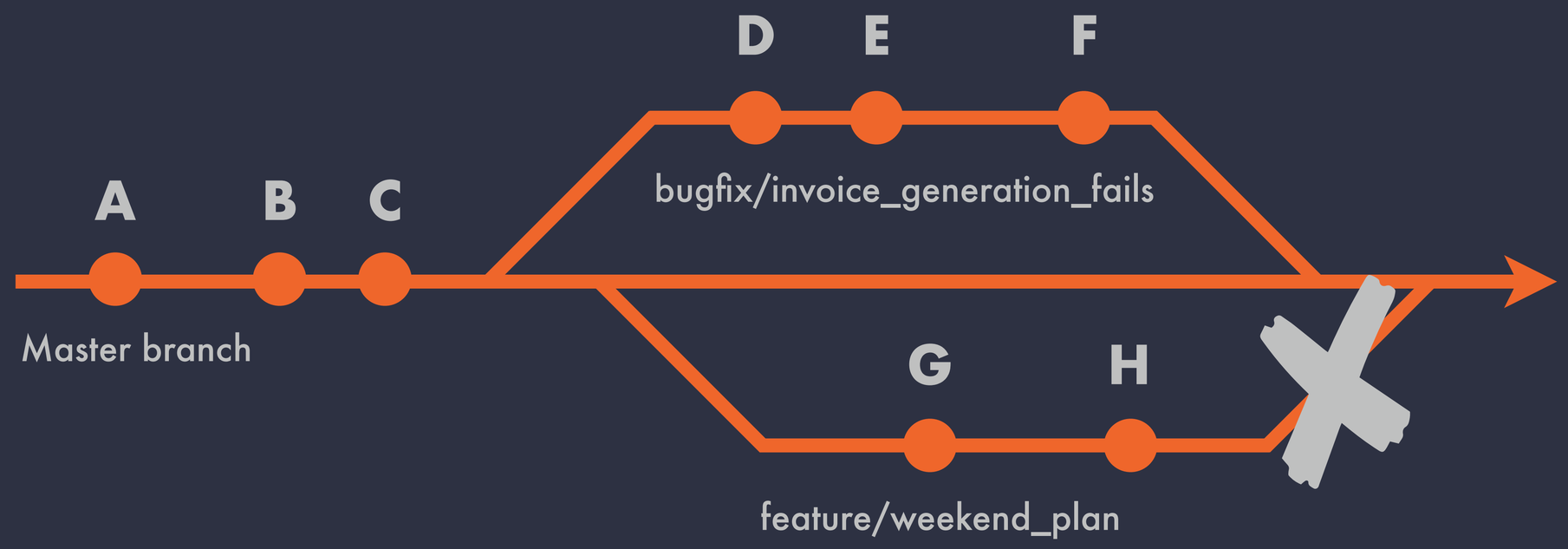
Index
Staging area

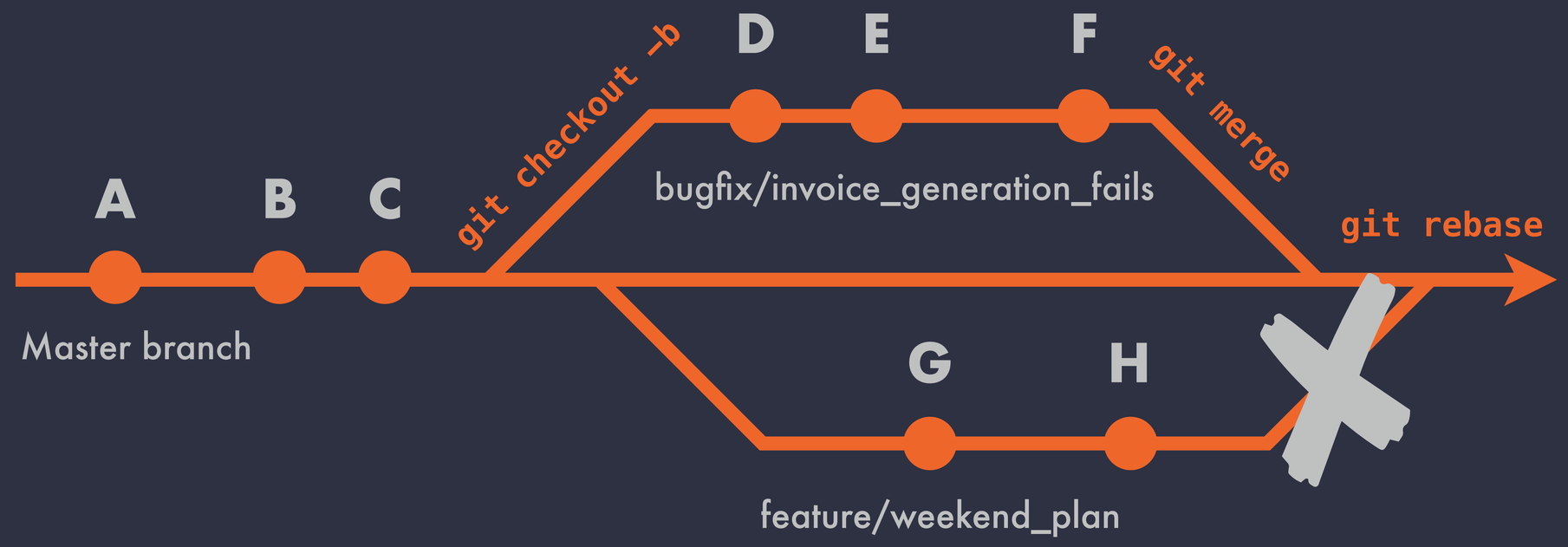
Local repository

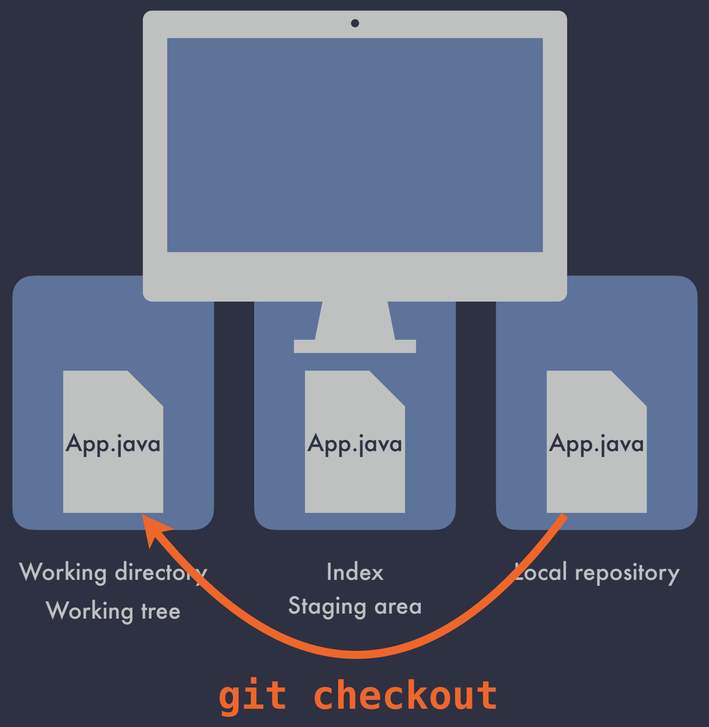


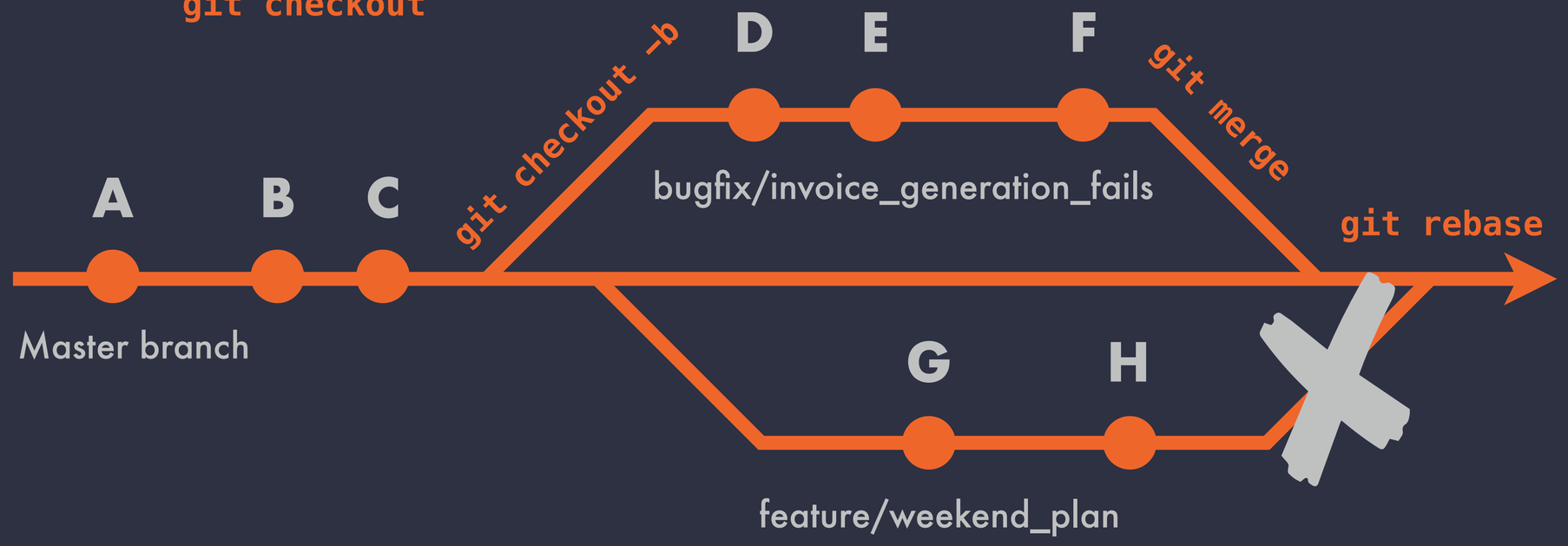
Remote repository

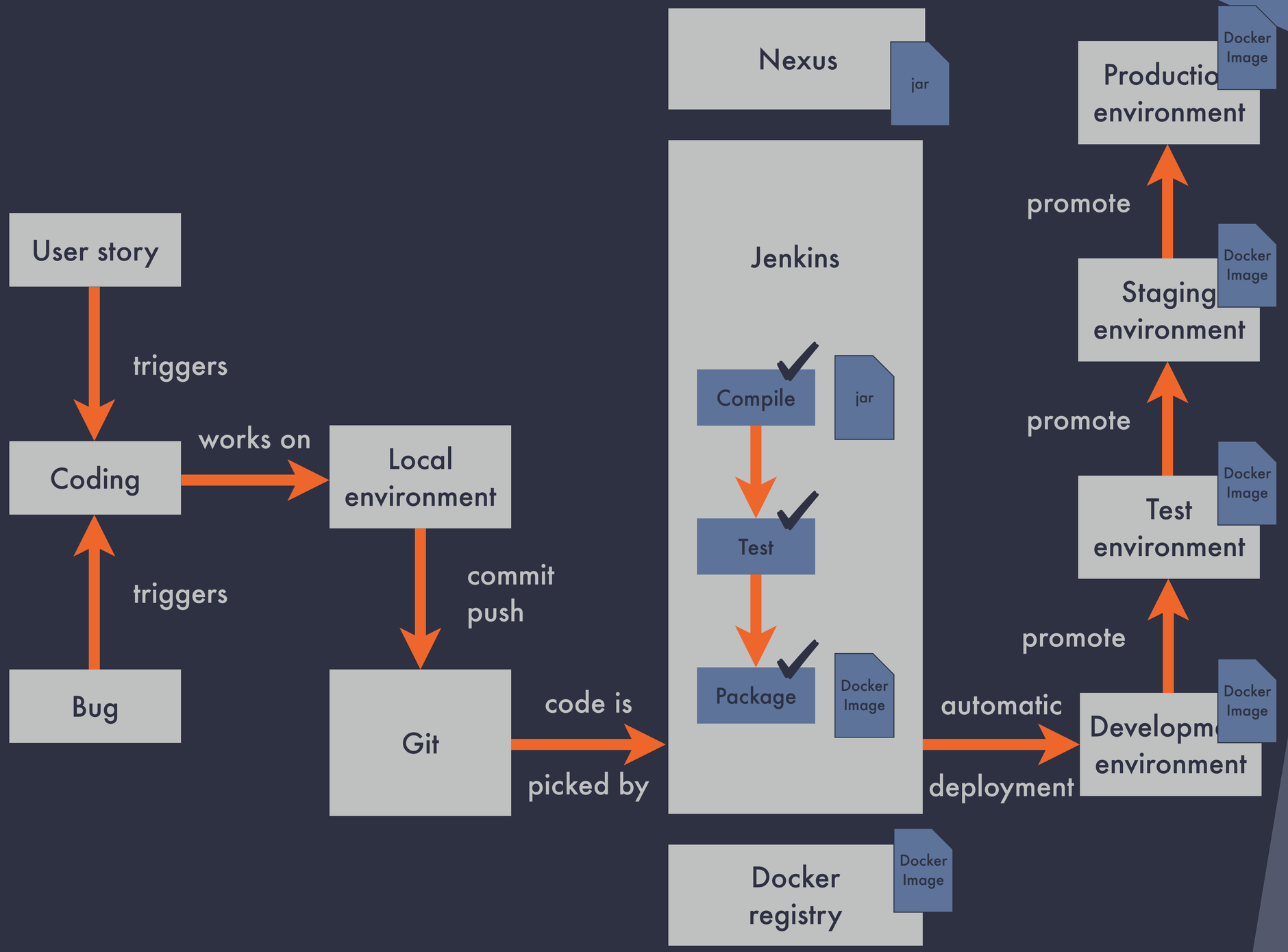

git init
git clone
git add
git commit
git push
App.java
App.java
git merge
git pull
git fetch
Working directory
Working tree
Index
Staging area
Local repository
Remote repository
git status


D
E
F
A
B
C
bugfix/invoice_generation_fails
Master branch
G
H
feature/weekend_plan


A
B
C
git checkout -b
D
E
F
git merge
bugfix/invoice_generation_fails
git rebase
Master branch
G
H
feature/weekend_plan


App.java
App.java
App.java
Working directory
Working tree
Index
Staging area
Local repository
git checkout


A
B
C
git checkout -b
D
E
F
git merge
bugfix/invoice_generation_fails
git rebase
Master branch
G
H
feature/weekend_plan


User story
triggers
Coding
works on
Local environment
triggers
Bug
commit
push
Git
code is
picked by
Nexus
jar
Jenkins
Compile
jar
Test
Package
Docker Image
Docker registry
Docker Image
automatic
deployment
Development environment
Docker Image
promote
Test environment
Docker Image
promote
Staging environment
Docker Image
promote
Production environment
Docker Image

Gitflow Workflow

Vincent Driessen

https://nvie.com/posts/a-successful-git-branching-model/

Trunk Based Development

https://trunkbaseddevelopment.com/

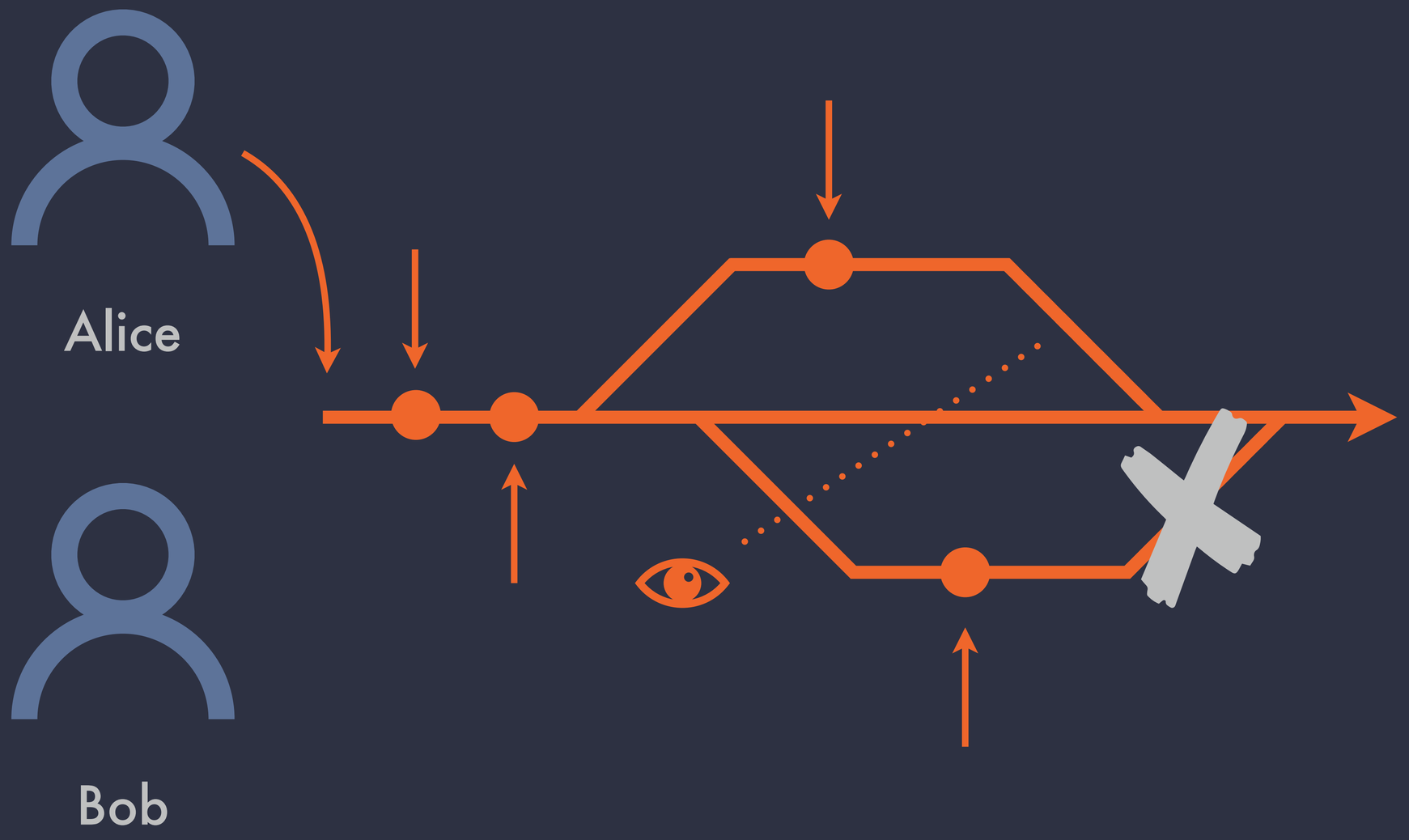
https://martinfowler.com/bliki/FeatureToggle.html

Demo

GitHub

Collaboration platform

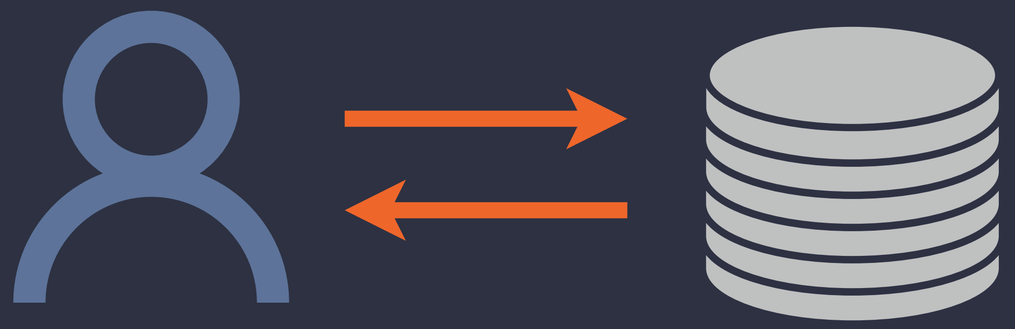
Git repository


Alice
Bob

Demo

Databases

Databases



Databases

Databases

Relational

RDBMS (Relational Database Management Systems)

Oracle (1979)

Databases

Relational

RDBMS (Relational Database Management Systems)

Oracle (1979), IBM DB2 (1983), MS SQL Server (1989),
MySQL (1995), PostgreSQL (1996)

Base relation ≈ table

Databases

Relational

RDBMS (Relational Database Management Systems)

Oracle (1979), IBM DB2 (1983), MS SQL Server (1989),
MySQL (1995), PostgreSQL (1996)

Base relation ≈ table Derived relation ≈ view, query

SQL (Structured Query Language)
access, manipulate, define data
control permissions and transactions

Require schema

Provide data safety guarantees (ACID)

Scale up (vertical scaling)

Databases

Relational	NoSQL
RDBMS (Relational Database Management Systems)	No SQL, Not only SQL
Oracle (1979), IBM DB2 (1983), MS SQL Server (1989), MySQL (1995), PostgreSQL (1996)	CouchDB (2005), Cassandra (2008), HBase (2008), MongoDB (2009), Redis (2009)
Base relation ≈ table Derived relation ≈ view, query	Non-relational Key-value, document, wide-column stores
SQL (Structured Query Language) access, manipulate, define data control permissions and transactions	Do not support SQL
Require schema	Schema-less
Provide data safety guarantees (ACID)	Focus on performance and scalability
Scale up (vertical scaling)	Scale out (horizontal scaling)

invoice_id_seq

Sequence

Row
= Record
= Tuple

id	number	total
bigint	character varying(255)	numeric(11, 2)
1	INV/001	100.01
2	INV/002	200.02
3	INV/003	300.03

invoice

Noun in singular form

Relation
= Table

Not NULL
Primary Key

Column
= Attribute
= Field

Not NULL
Unique

number_idx

Index

invoice_id_idx

Index

id	description	total	invoice_id
bigint	character varying(255)	numeric(11, 2)	bigint
1	Day Rate	70.01	1
2	Night Rate	30.00	1
3	Stable Rate	200.02	2

invoice_line

Not NULL
Primary Key

Not NULL

Not NULL
Foreign Key

Constraints

SQL

Structured Query Language

INSERT
UPDATE
DELETE

SELECT

SELECT

```
select * from invoice

select number, total from invoice

select * from invoice i join invoice_line il on il.invoice_id = i.id
```

SELECT

```
select * from invoice

select number, total from invoice

select * from invoice i inner join invoice_line il on il.invoice_id = i.id
```

SELECT

```
select * from invoice

select number, total from invoice

select * from invoice i join invoice_line il on il.invoice_id = i.id

select * from invoice i left join invoice_line il on il.invoice_id = i.id
```

SELECT

```
select * from invoice

select number, total from invoice

select * from invoice i join invoice_line il on il.invoice_id = i.id

select * from invoice i left outer join invoice_line il on il.invoice_id = i.id
```

SELECT

```
select * from invoice

select number, total from invoice

select * from invoice i join invoice_line il on il.invoice_id = i.id

select * from invoice i left join invoice_line il on il.invoice_id = i.id
```

invoice

id	number	total
bigint	character varying(255)	numeric(11, 2)
1	INV/001	100.01
2	INV/002	200.02
3	INV/003	300.03

Normalised

invoice_line

id	description	total	invoice_id
bigint	character varying(255)	numeric(11, 2)	bigint
1	Day Rate	70.01	1
2	Night Rate	30.00	1
3	Stable Rate	200.02	2

Denormalised

id	description	total	invoice_number	invoice_total
bigint	character varying(255)	numeric(11, 2)	character varying(255)	numeric(11, 2)
1	Day Rate	70.01	INV/001	100.01
2	Night Rate	30.00	INV/001	100.01
3	Stable Rate	200.02	INV/002	200.02

invoice_line

invoice

id	number	total
bigint	character varying(255)	numeric(11, 2)
1	INV/001	100.01
2	INV/002	200.02
3	INV/003	300.03

invoice_line

id	description	total	invoice_id
bigint	character varying(255)	numeric(11, 2)	bigint
1	Day Rate	70.01	1
2	Night Rate	30.00	1
3	Stable Rate	200.02	2

invoice_line

id	description	total	invoice_number	invoice_total
bigint	character varying(255)	numeric(11, 2)	character varying(255)	numeric(11, 2)
1	Day Rate	70.01	INV/001	100.01
2	Night Rate	30.00	INV/001	100.01
3	Stable Rate	200.02	INV/002	200.02

SELECT

```
select * from invoice

select number, total from invoice

select * from invoice i join invoice_line il on il.invoice_id = i.id

select * from invoice i left join invoice_line il on il.invoice_id = i.id
```

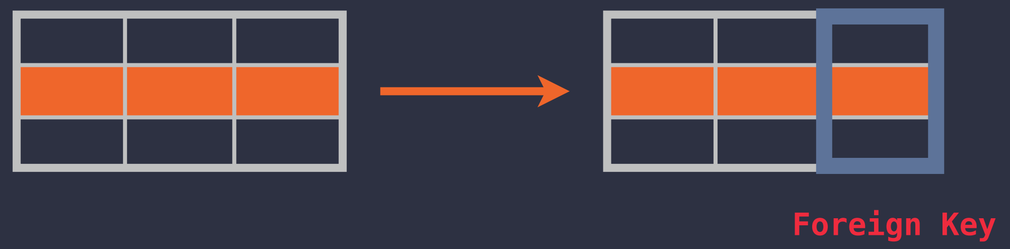
one-to-one

one-to-many

many-to-one

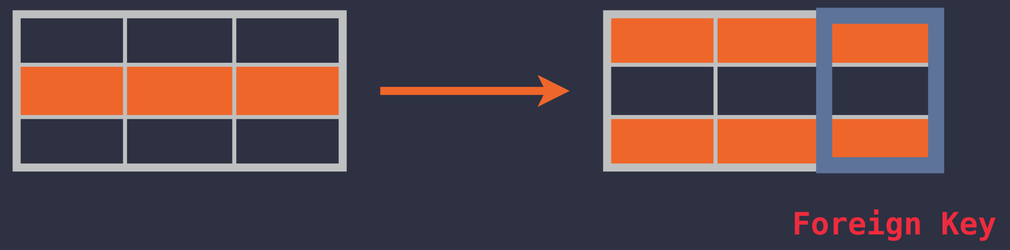
many-to-many

one-to-one



one-to-many

many-to-one

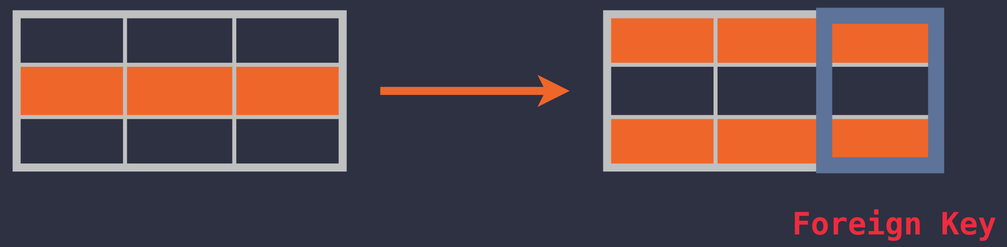


many-to-many

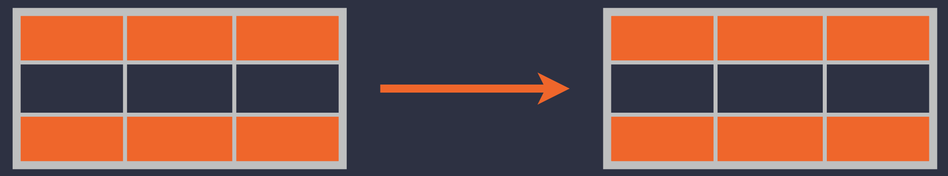
one-to-one

one-to-many

many-to-one



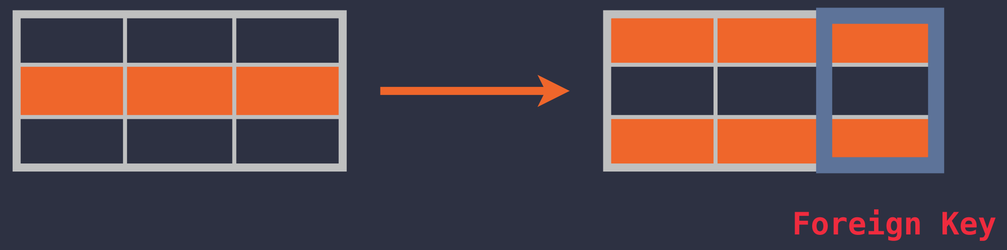
many-to-many



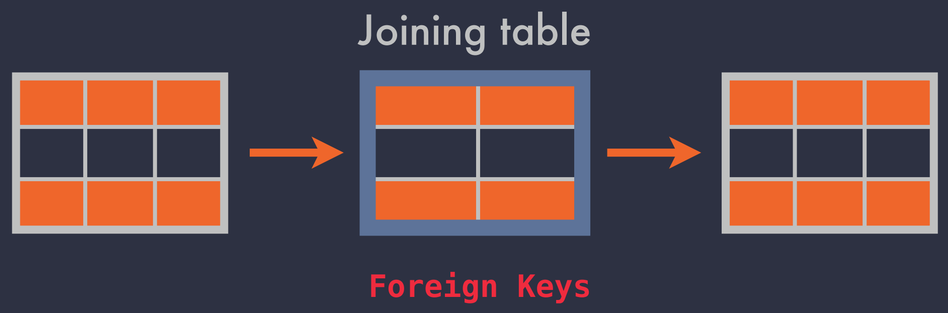
one-to-one

one-to-many

many-to-one



many-to-many



Transactions

Transactions

Transaction = single logical operation

Atomicity
Consistency
Isolation
Durability

Atomicity

SELECT... → INSERT... → UPDATE... → UPDATE...

Consistency

SELECT... → INSERT... → UPDATE... → UPDATE...

Not NULL
Unique
Foreign Key

Isolation

SELECT... → INSERT... → UPDATE... → UPDATE...

SELECT... → INSERT... → UPDATE... → UPDATE...

Durability

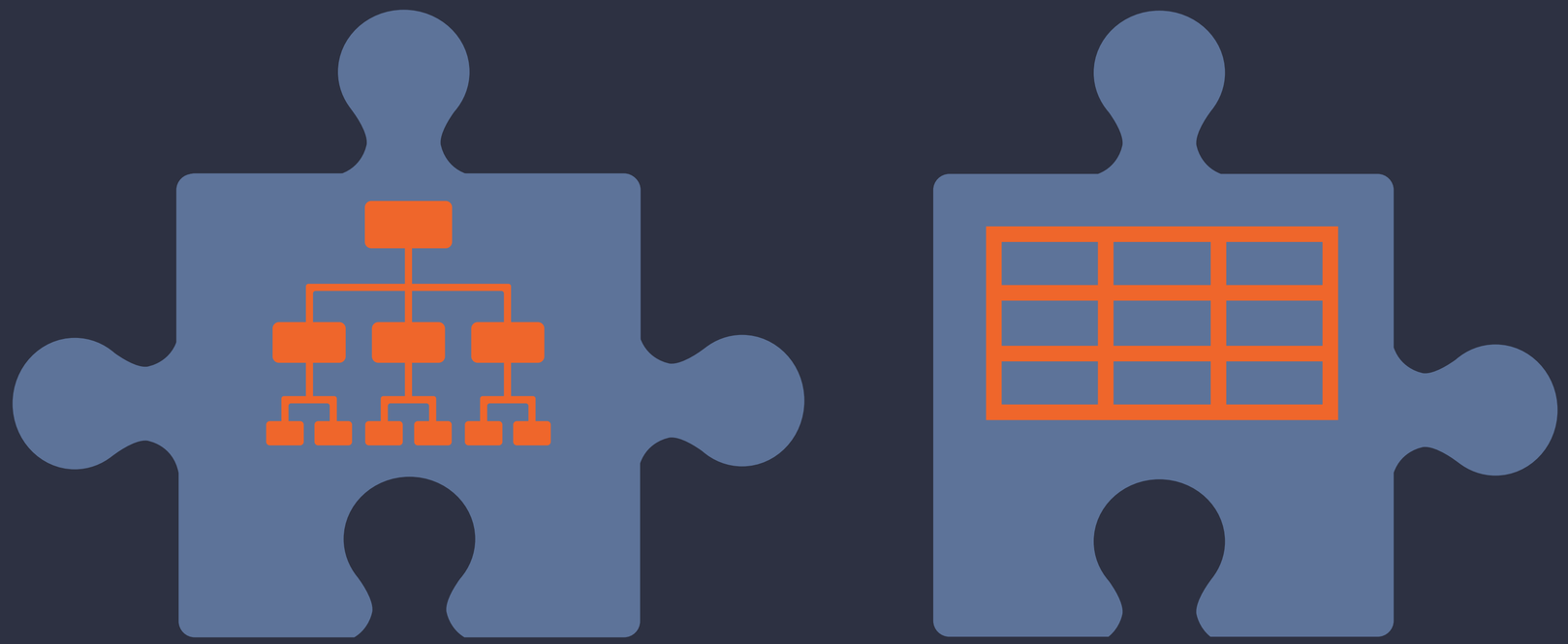
SELECT... → INSERT... → UPDATE... → UPDATE...

Demo

Relational Databases and Web Applications

Object-relational mismatch

A.k.a.: impedance mismatch

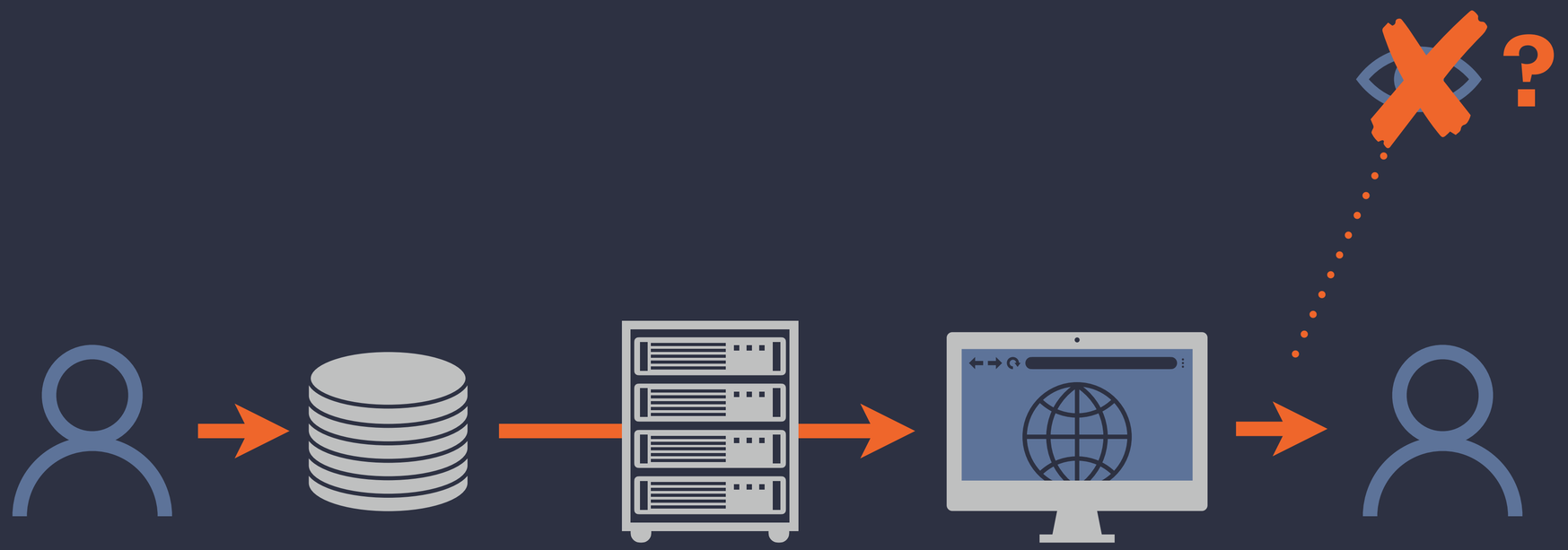


ORM

Object-Relational Mapping

application.yml

```yaml
spring:
  application:
    name: nrgyinvoicr
  servlet:
    multipart:
      max-file-size: 170MB
      max-request-size: 170MB
  datasource:
    url: jdbc:postgresql://localhost:5432/nrgyinvoicr
    username: nrgyinvoicr
    password: nrgyinvoicr
  jpa:
    generate-ddl: false
    properties:
      hibernate:
        jdbc:
          batch_size: 500
        order_inserts: true
        order_updates: true
  liquibase:
    change-log: classpath:db/changelog/db.changelog-master.xml
```



Cache

application.yml

```yaml
spring:
  application:
    name: nrgyinvoicr
  servlet:
    multipart:
      max-file-size: 170MB
      max-request-size: 170MB
  datasource:
    url: jdbc:postgresql://localhost:5432/nrgyinvoicr
    username: nrgyinvoicr
    password: nrgyinvoicr
  jpa:
    generate-ddl: false
    properties:
      hibernate:
        jdbc:
          batch_size: 500
        order_inserts: true

  liquibase:
    change-log: classpath:db/changelog/db.changelog-master.xml
```

https://www.liquibase.org

Tips on Learning Databases

Joins, aggregate functions

Filtering data, subqueries

Practice a lot!

Data types

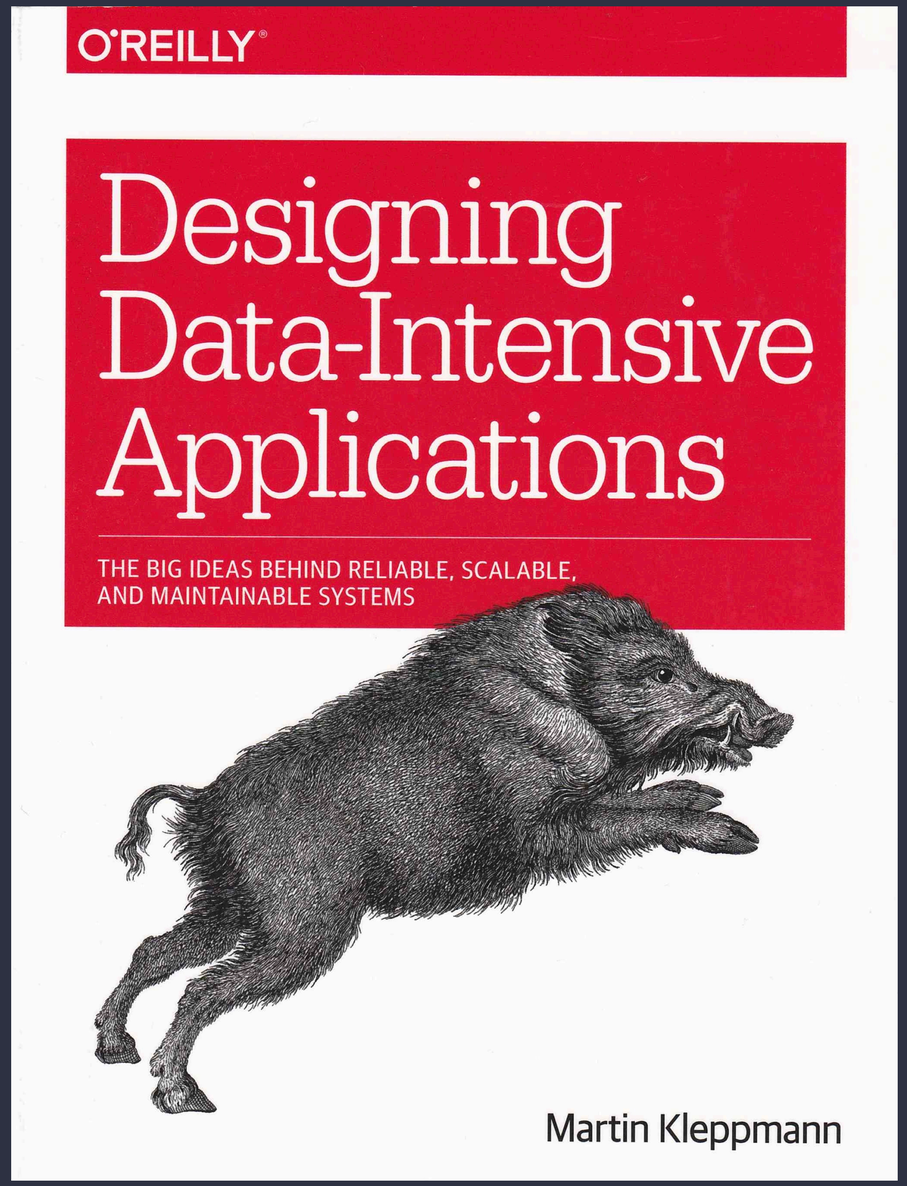
https://www.postgresql.org/docs/current/datatype.html

Indexes

Isolation levels

MVCC (MultiVersion Concurrency Control)

Optimistic vs pessimistic locking


O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann

Network and Infrastructure

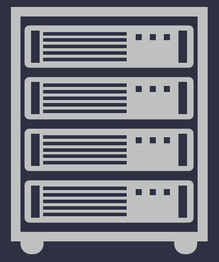
Network and Infrastructure

localhost:8080

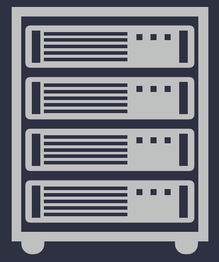
localhost:4200

127.0.0.1

localhost:5432



Application Server



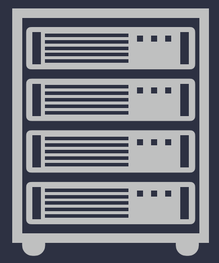
Angular Live Development Server

Database

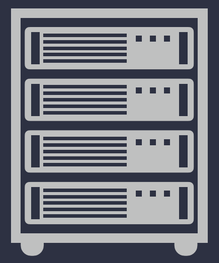
localhost:8080

localhost:4200

localhost:5432



Application Server

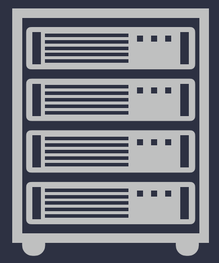


Angular Live Development Server

Database

http://localhost:4200

localhost:8080

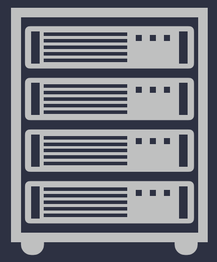


Application Server

Database

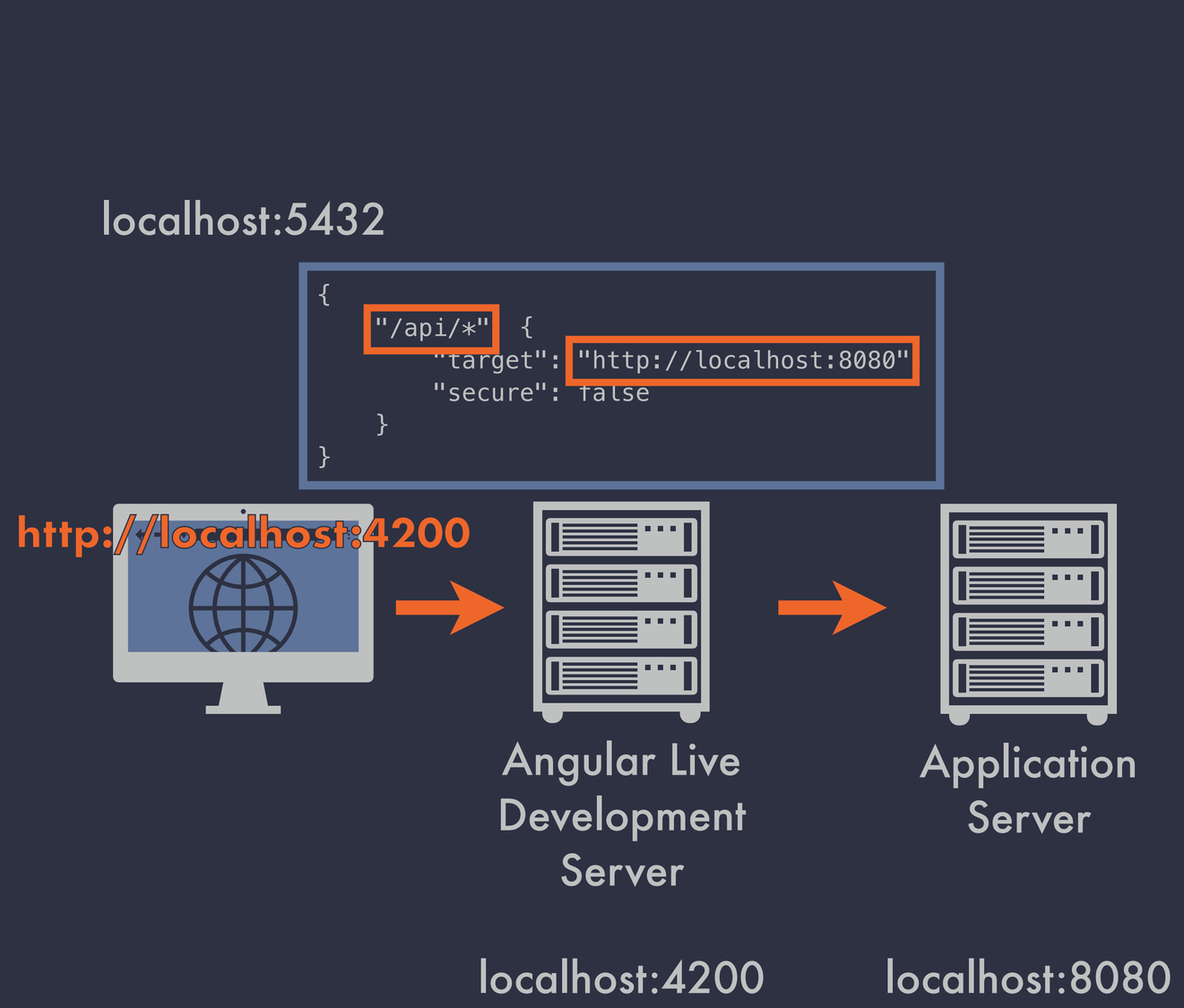
localhost:5432

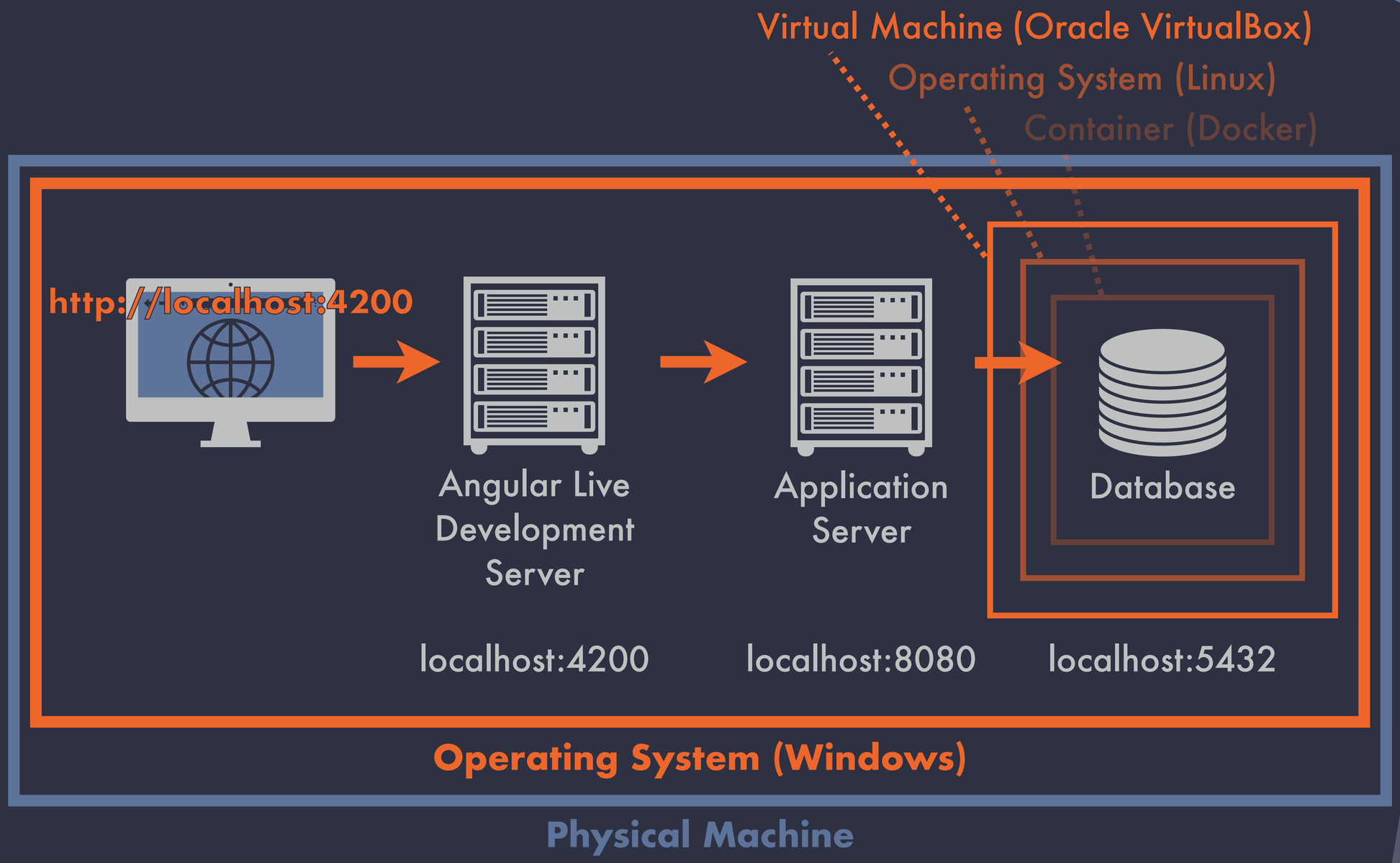
http://localhost:4200



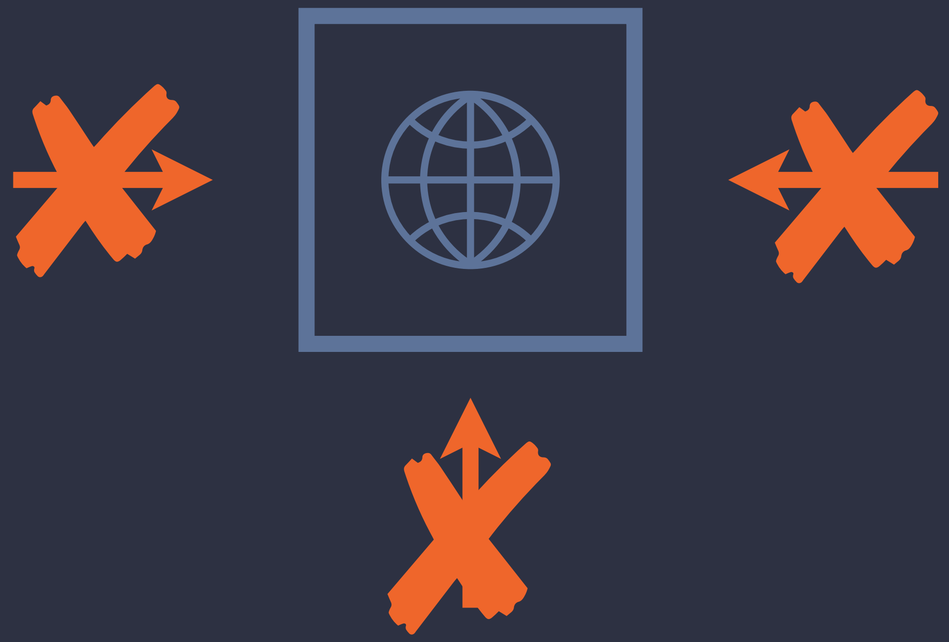
Angular Live Development Server

localhost:4200

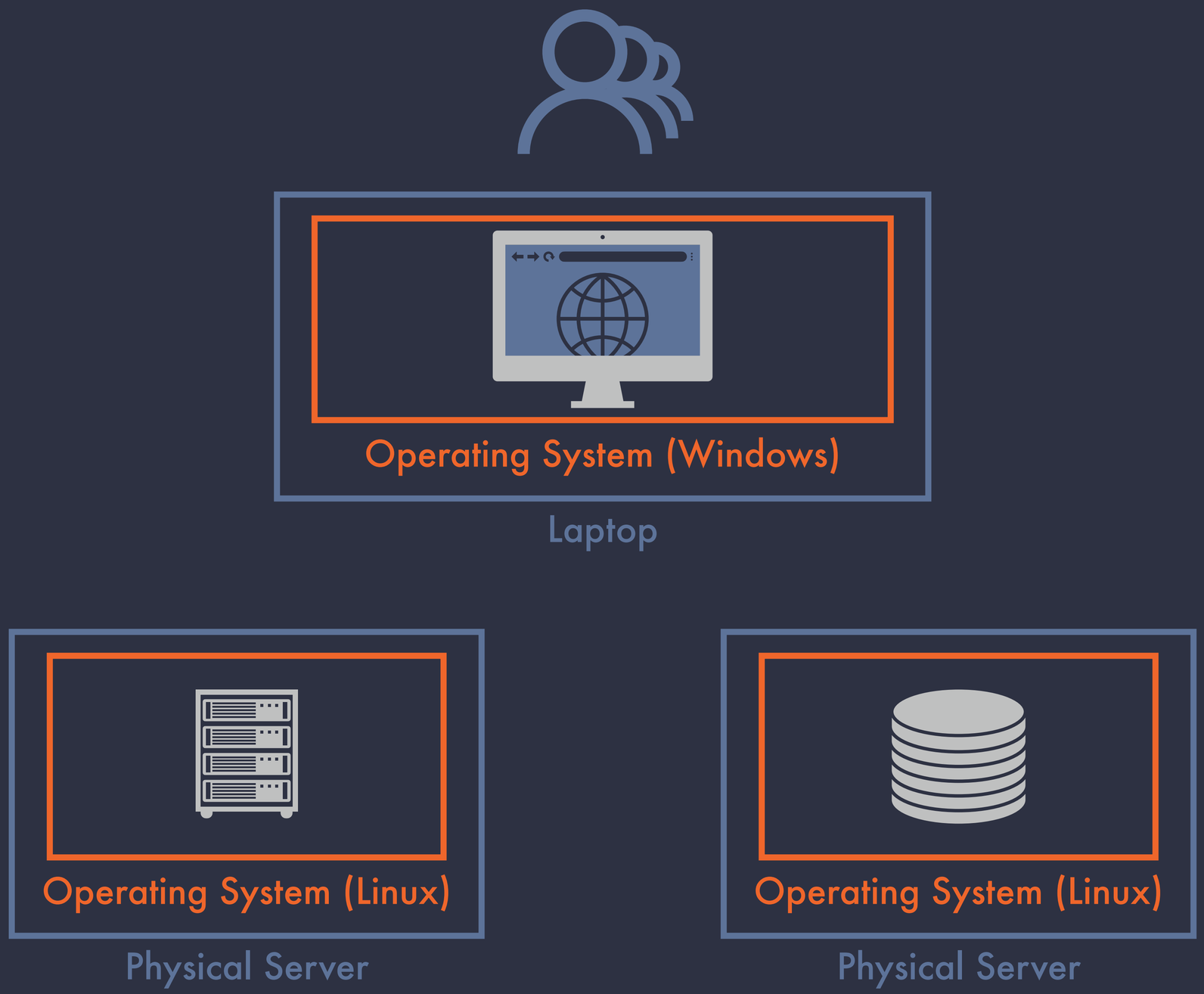

localhost:5432
Database
{
"/api/*" {
"target": "http://localhost:8080"
"secure": false
}
}
http://localhost:4200
Angular Live Development Server
Application Server
localhost:4200
localhost:8080

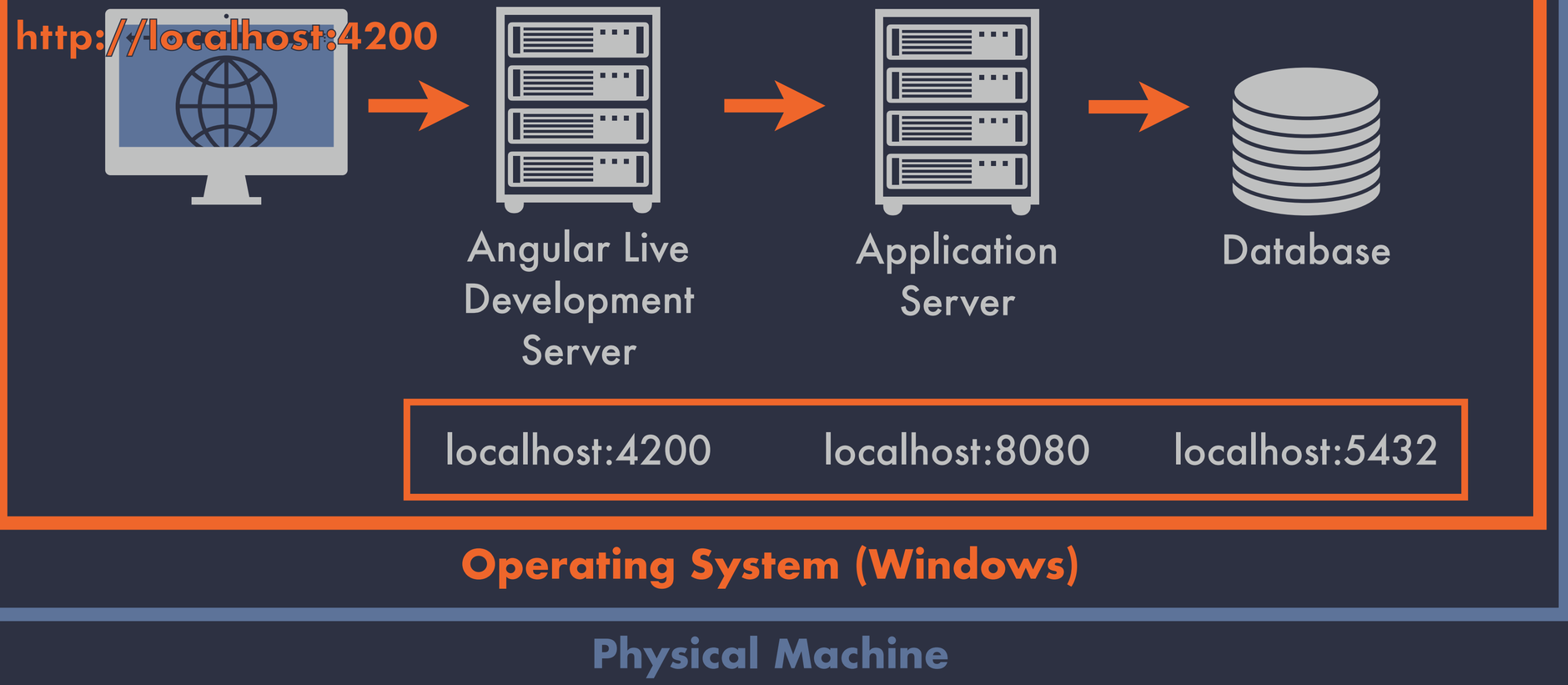

Virtual Machine (Oracle VirtualBox)
Operating System (Linux)
Container (Docker)
http://localhost:4200
Angular Live Development Server
Application Server
Database
localhost:4200
localhost:8080
localhost:5432
Operating System (Windows)
Physical Machine

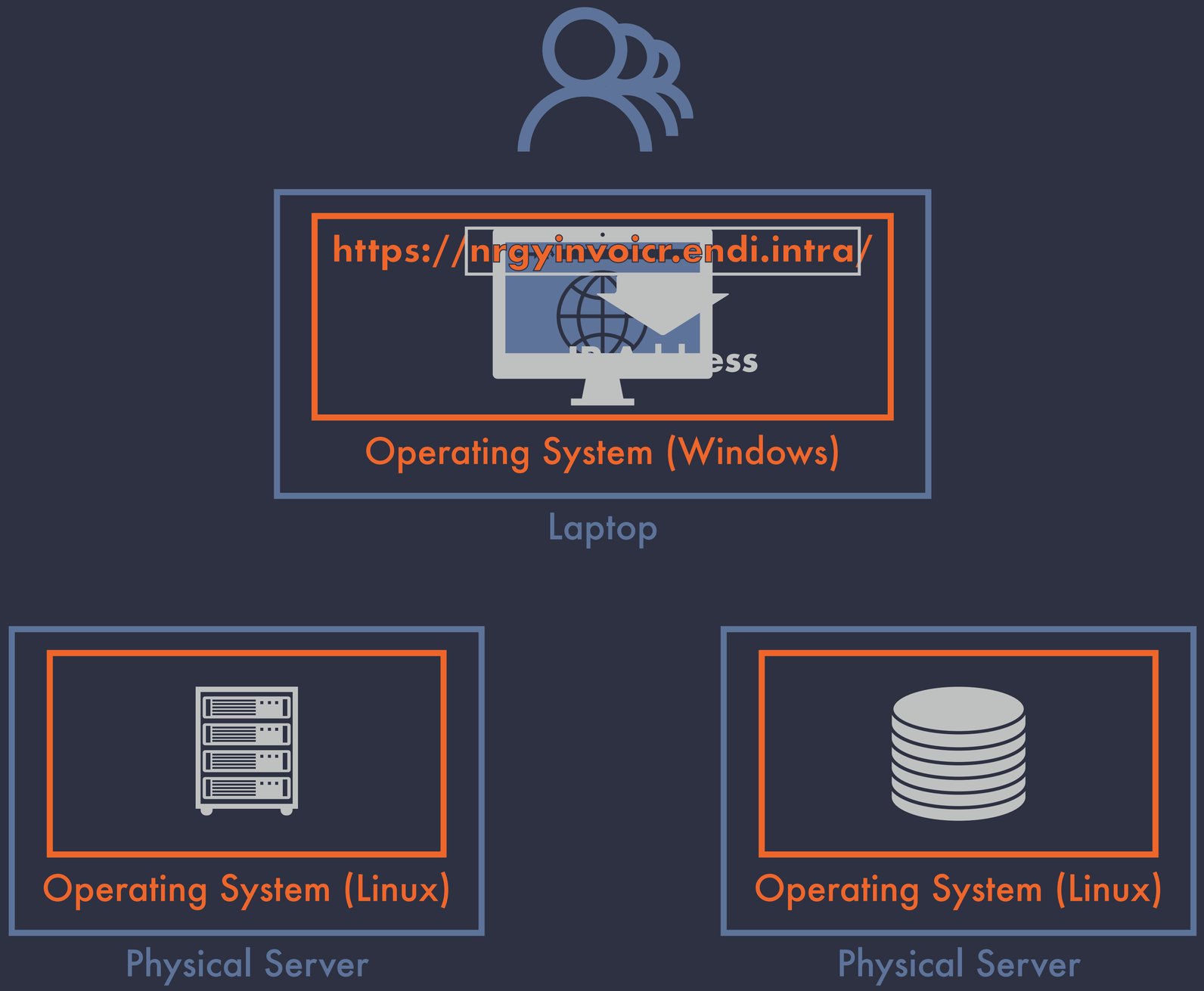
Intranet

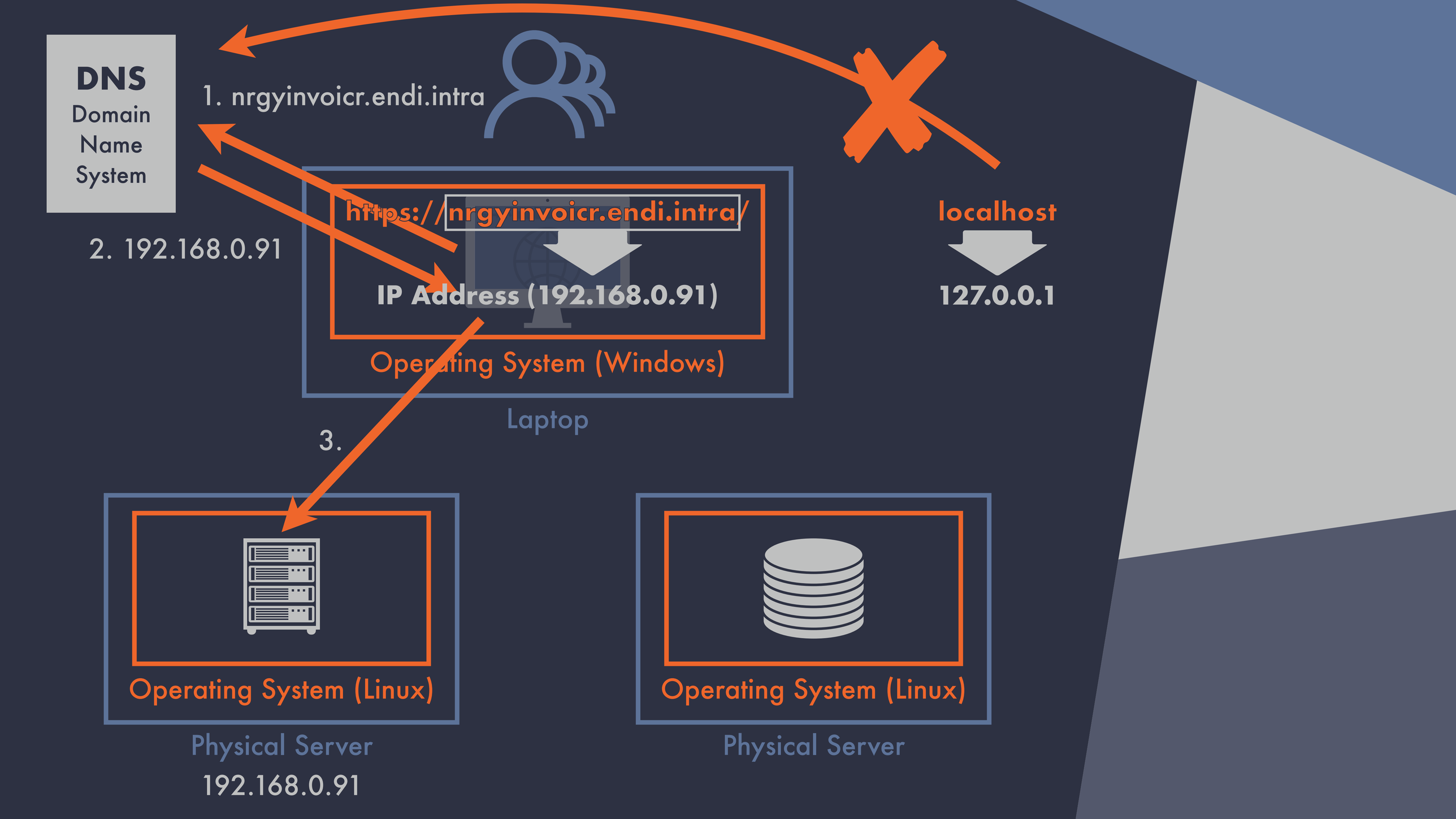


Internet


Operating System (Windows)
Laptop
Operating System (Linux)
Physical Server
Operating System (Linux)
Physical Server


http://localhost:4200
Angular Live Development Server
Application Server
Database
localhost:4200
localhost:8080
localhost:5432
Operating System (Windows)
Physical Machine


https://nrgyinvoicr.endi.intra/
Operating System (Windows)
Laptop
Operating System (Linux)
Physical Server
Operating System (Linux)
Physical Server


DNS
Domain
Name
System
1. nrgyinvoicr.endi.intra
2. 192.168.0.91
https://nrgyinvoicr.endi.intra/
IP Address (192.168.0.91)
Operating System (Windows)
Laptop
localhost
127.0.0.1
3.
Operating System (Linux)
Physical Server
192.168.0.91
Operating System (Linux)
Physical Server

TCP

Transmission Control Protocol

IP

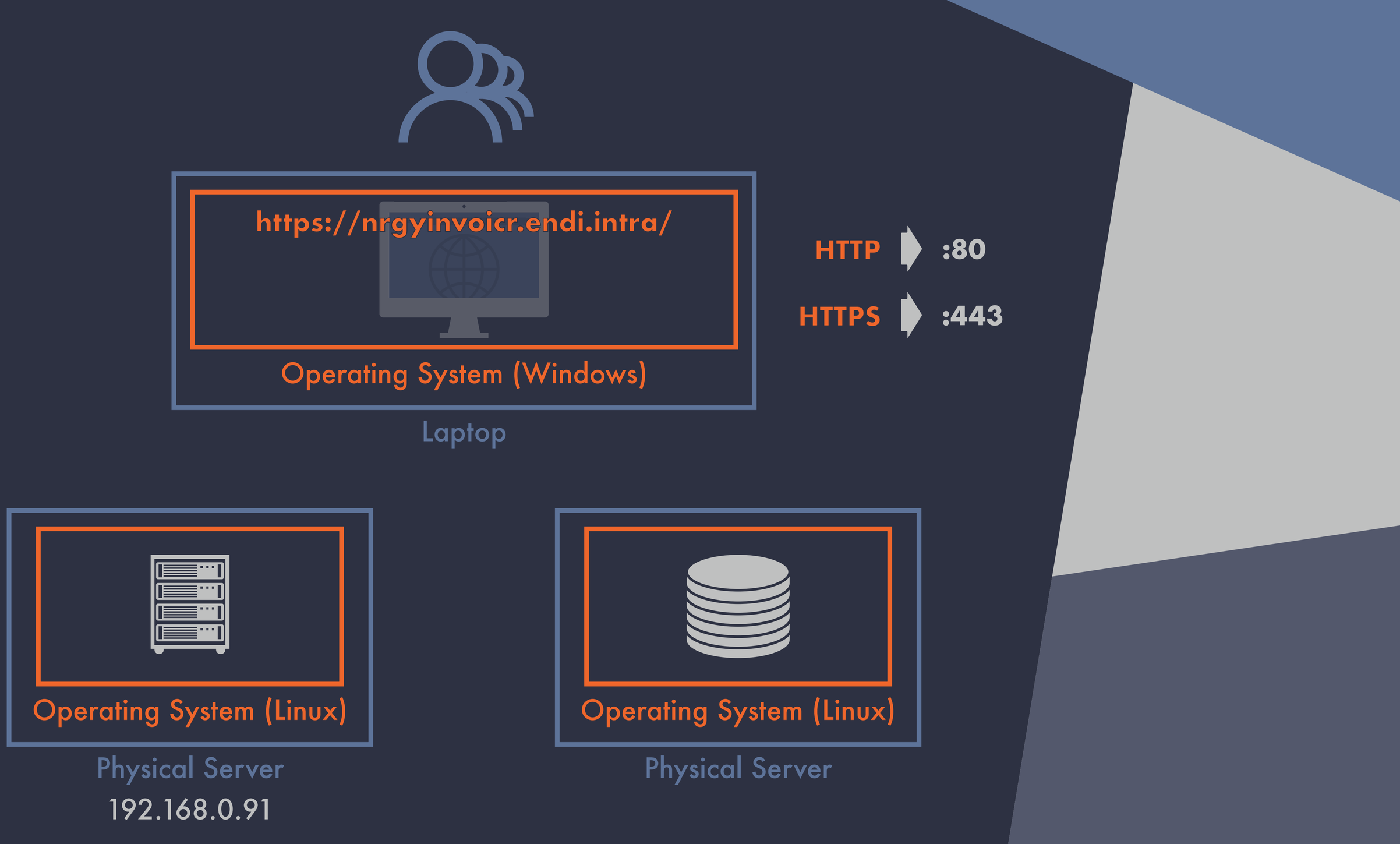
Internet Protocol

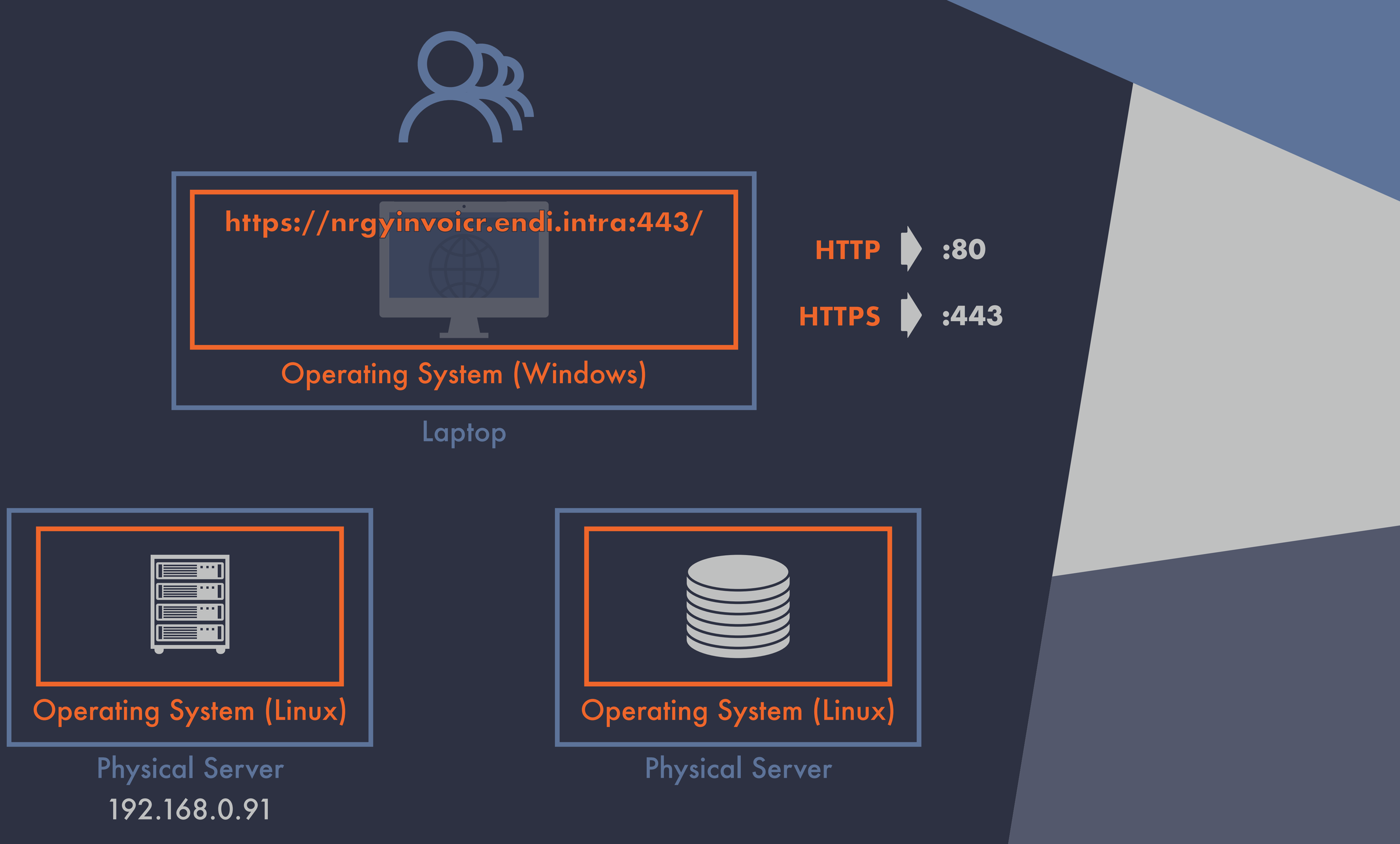
TCP/IP

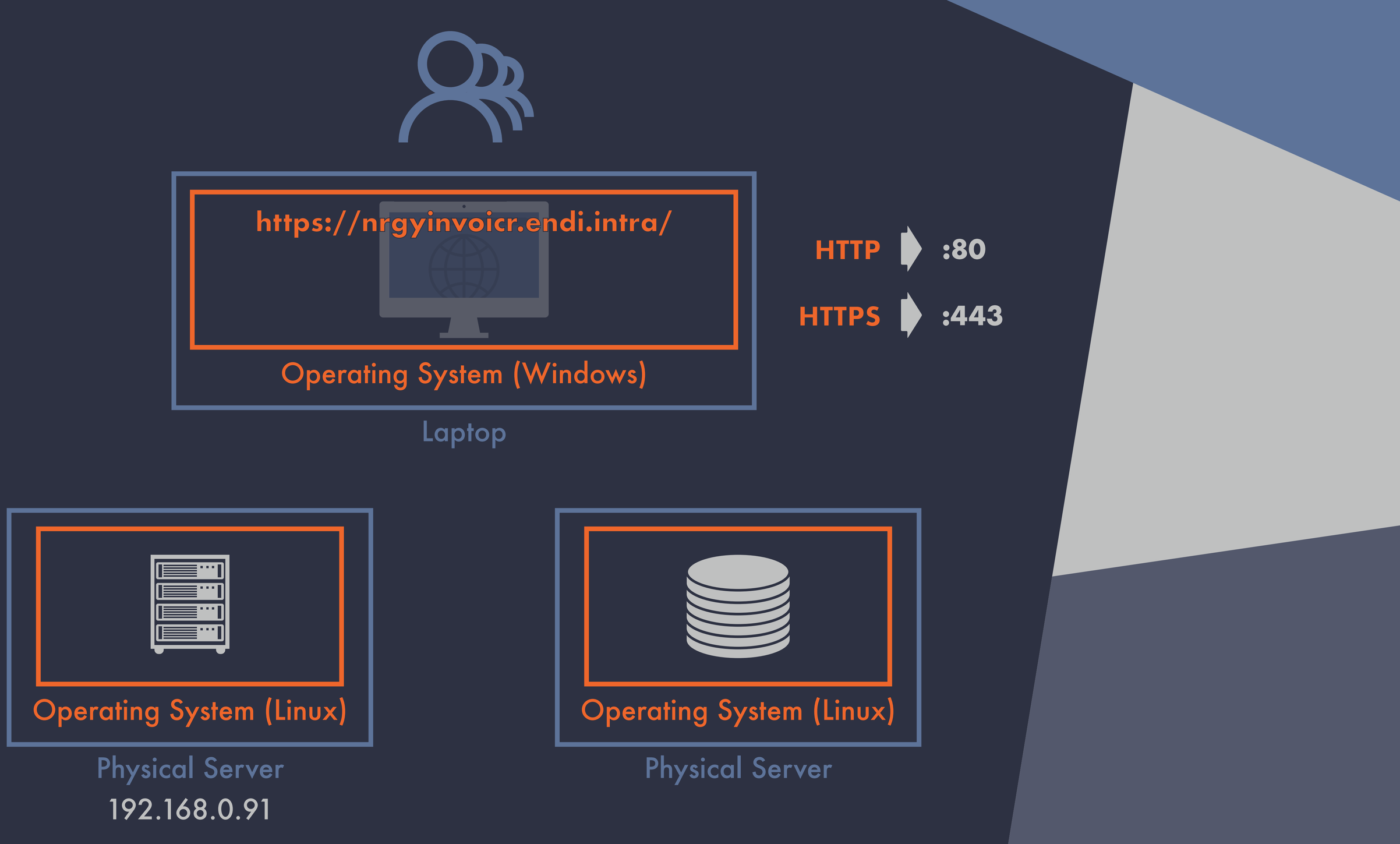
HTTP

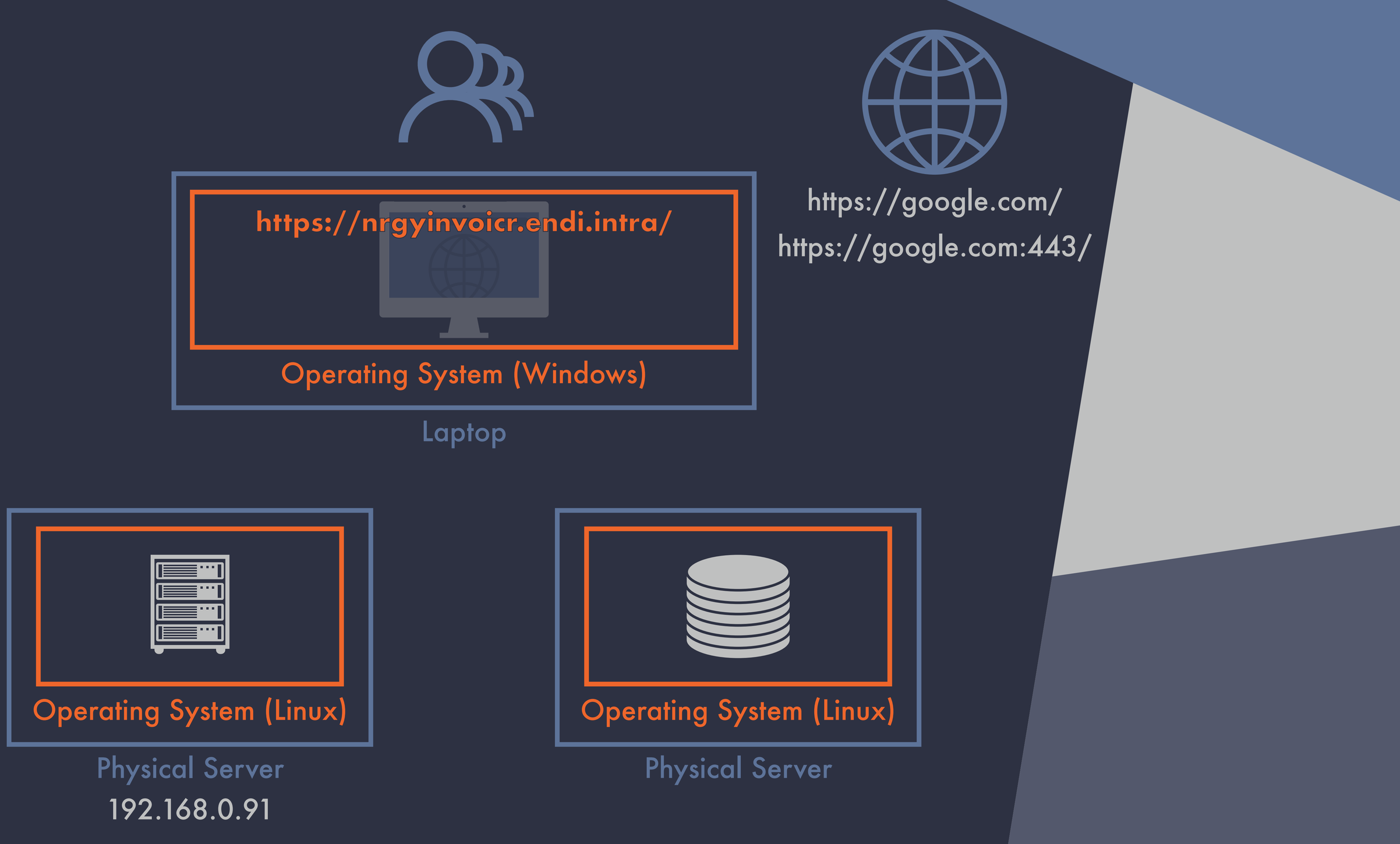
HyperText Transfer Protocol

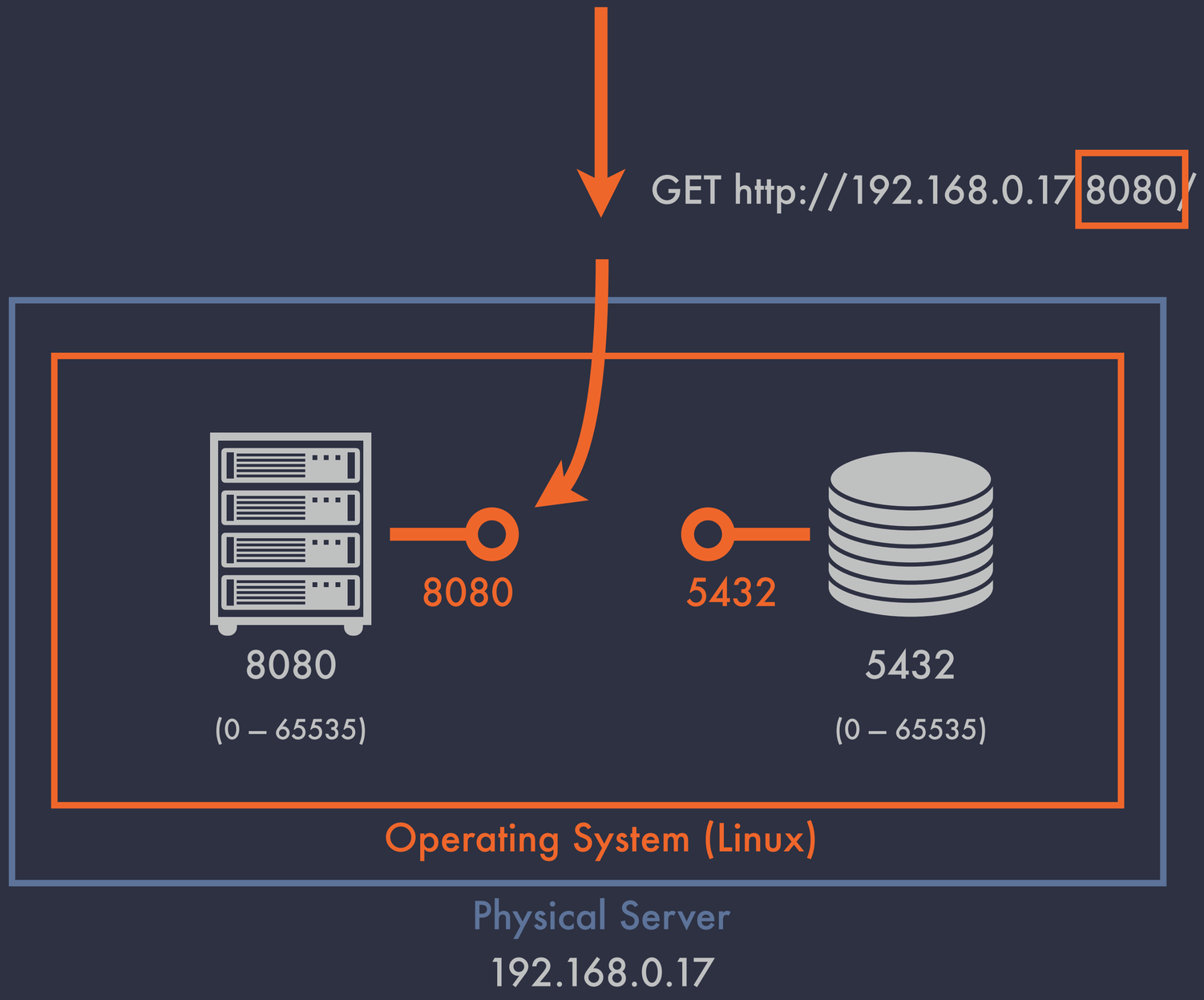
TCP/IP

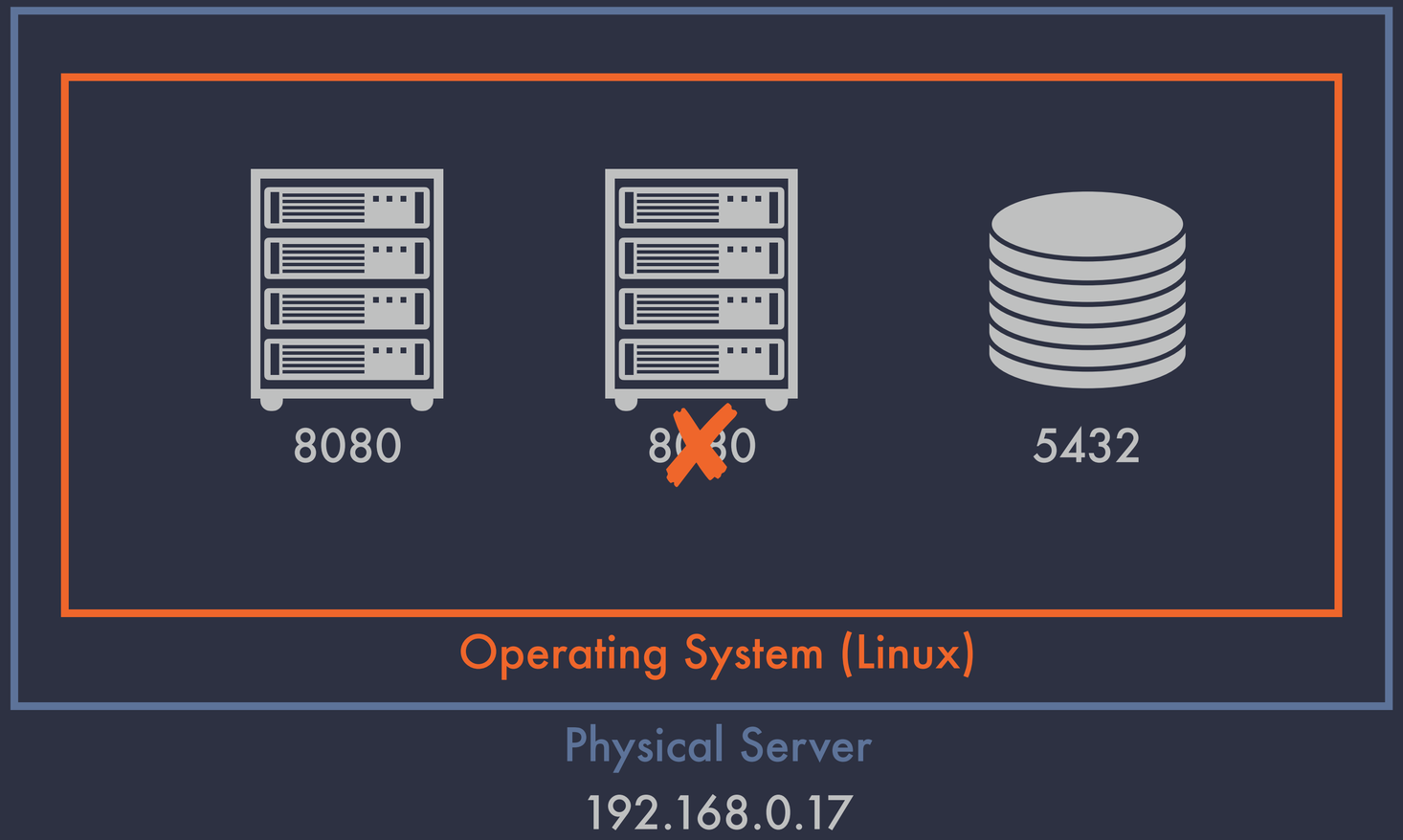

https://nrgyinvoicr.endi.intra/
Operating System (Windows)
Laptop
HTTP :80
HTTPS :443
Operating System (Linux)
Physical Server
192.168.0.91
Operating System (Linux)
Physical Server

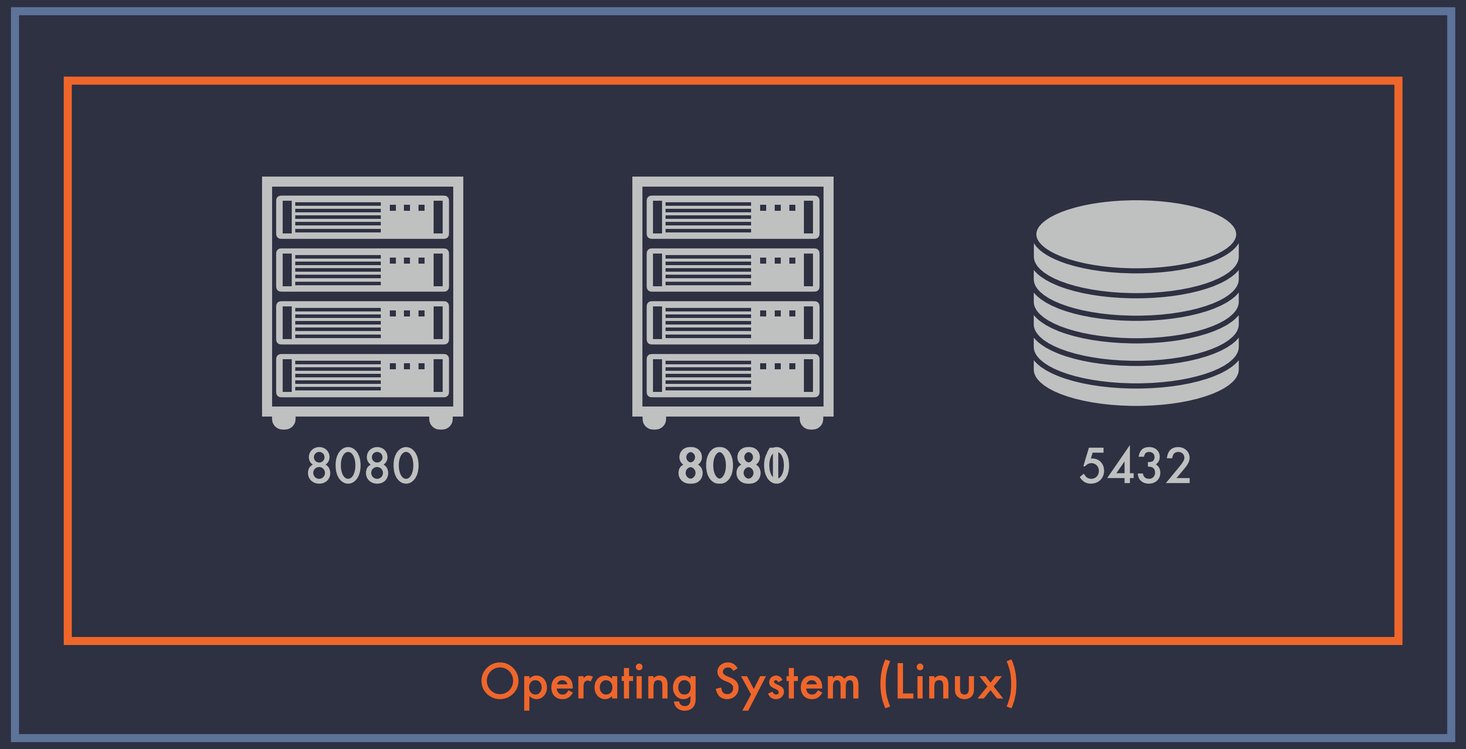

https://nrgyinvoicr.endi.intra:443/
Operating System (Windows)
Laptop
HTTP :80
HTTPS :443
Operating System (Linux)
Physical Server
192.168.0.91
Operating System (Linux)
Physical Server

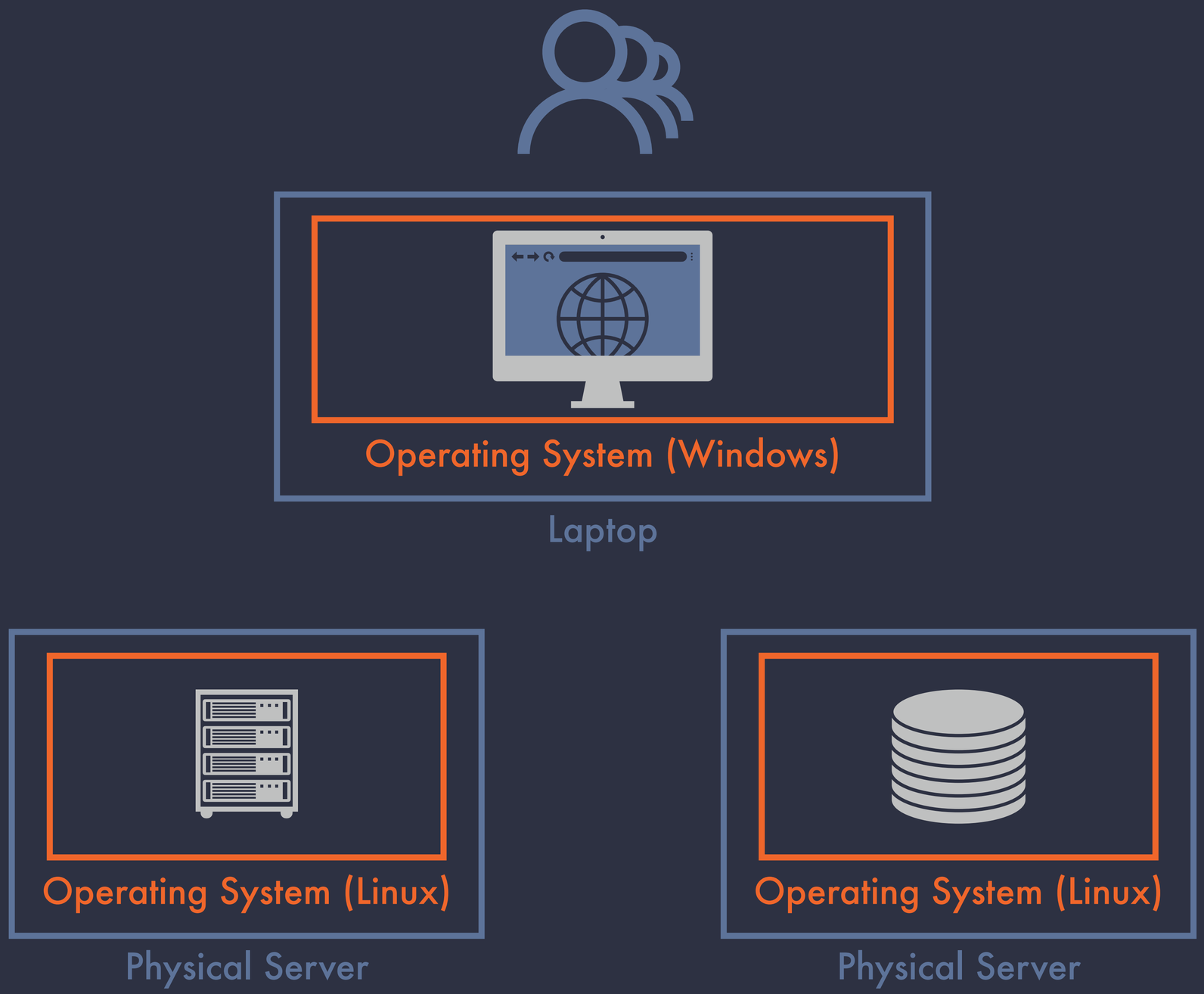

https://nrgyinvoicr.endi.intra/
Operating System (Windows)
Laptop
HTTP :80
HTTPS :443
Operating System (Linux)
Physical Server
192.168.0.91
Operating System (Linux)
Physical Server

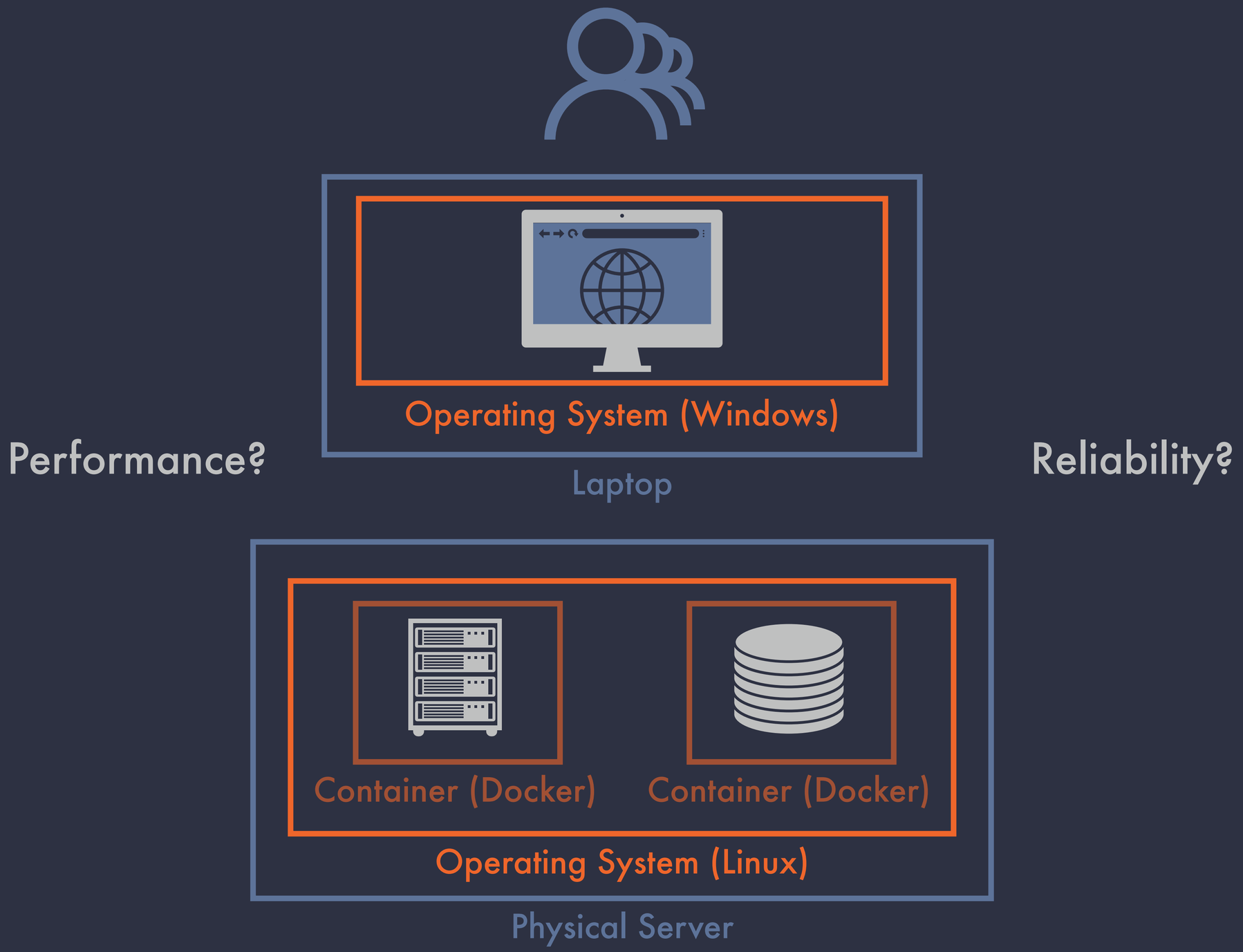

https://nrgyinvoicr.endi.intra/
Operating System (Windows)
Laptop
https://google.com/
https://google.com:443/
Operating System (Linux)
Physical Server
192.168.0.91
Operating System (Linux)
Physical Server

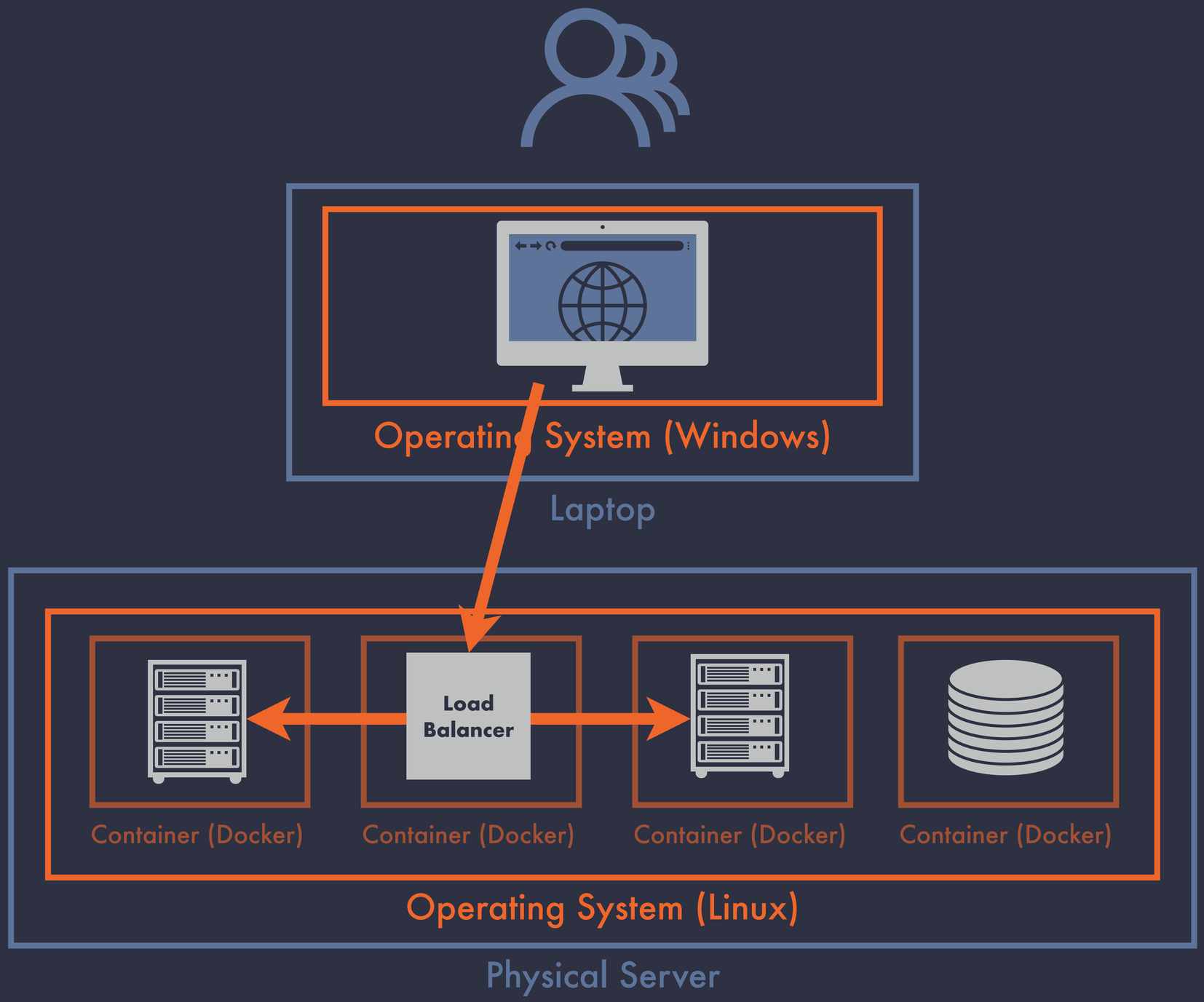

GET http://192.168.0.17 8080/
8080
8080
(0 – 65535)
5432
5432
(0 – 65535)
Operating System (Linux)
Physical Server
192.168.0.17

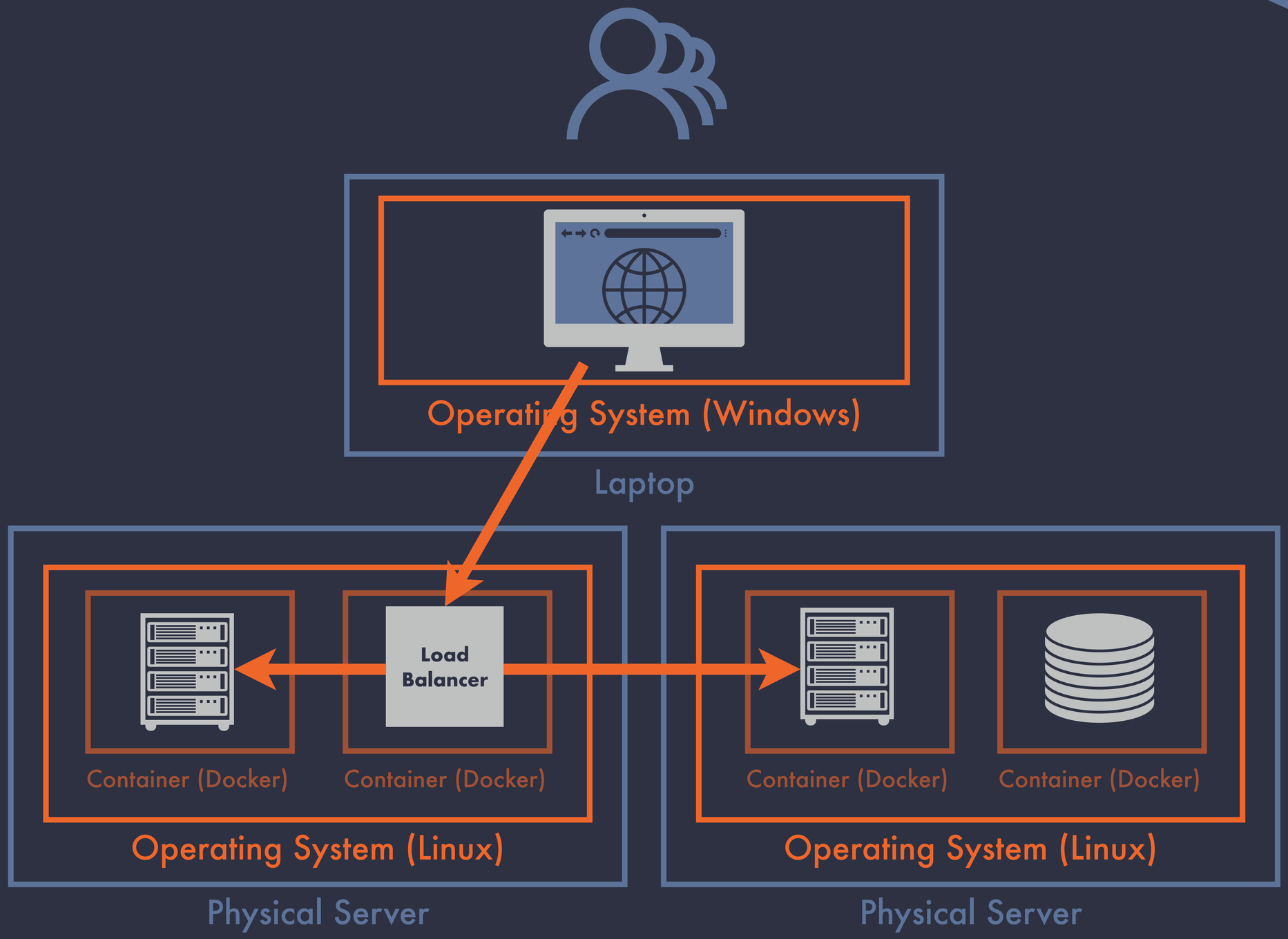

8080
8080
5432
Operating System (Linux)
Physical Server
192.168.0.17

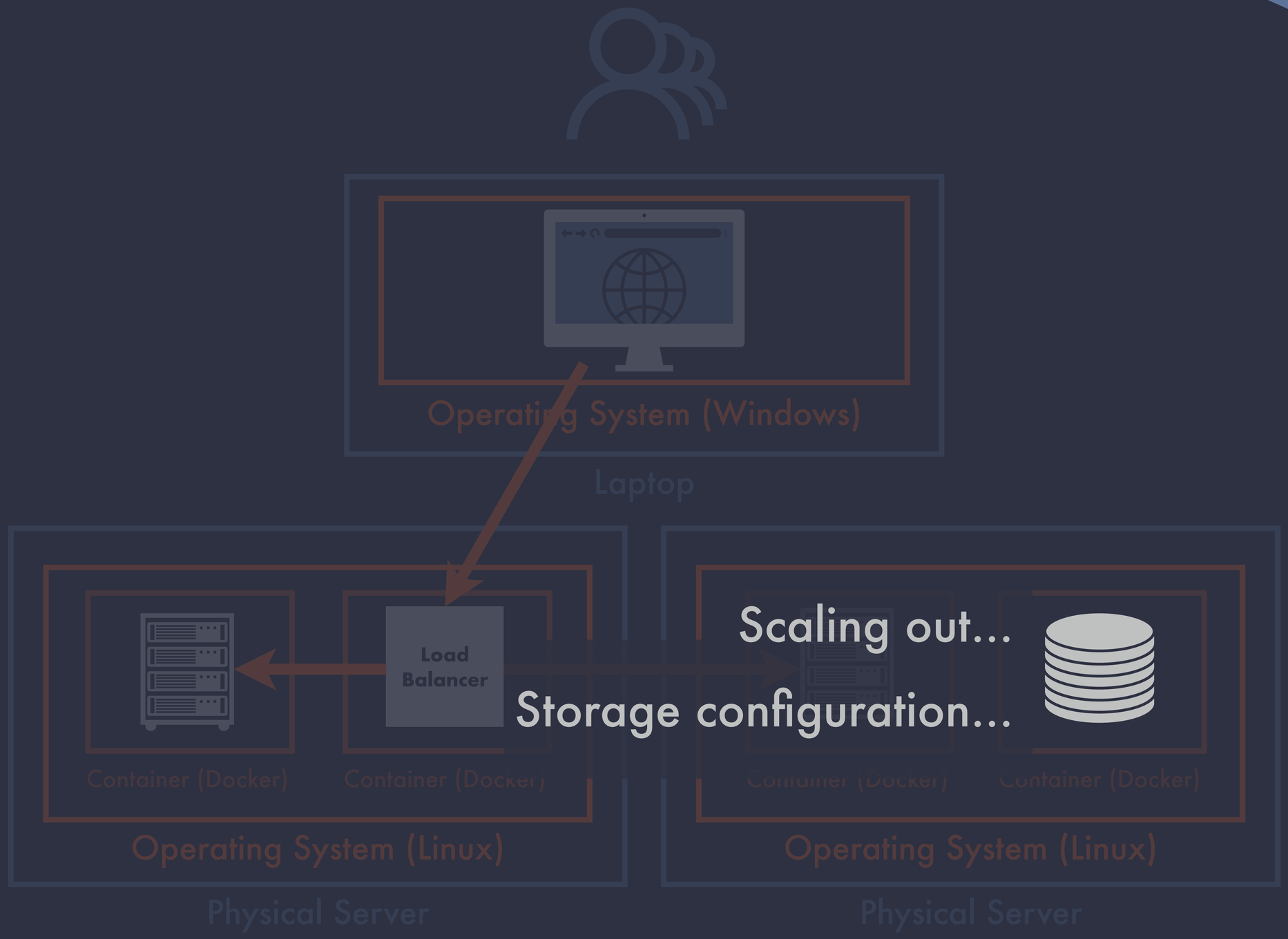

8080
8080
5432
Operating System (Linux)
Physical Server
192.168.0.17


Operating System (Windows)
Laptop
Operating System (Linux)
Physical Server
Operating System (Linux)
Physical Server


Operating System (Windows)
Laptop
Performance?
Reliability?
Container (Docker)
Container (Docker)
Operating System (Linux)
Physical Server


Operating System (Windows)
Laptop
Load Balancer
Container (Docker)
Container (Docker)
Container (Docker)
Container (Docker)
Operating System (Linux)
Physical Server


Operating System (Windows)
Laptop
Load Balancer
Container (Docker)
Container (Docker)
Operating System (Linux)
Physical Server
Container (Docker)
Container (Docker)
Operating System (Linux)
Physical Server



Scaling out…

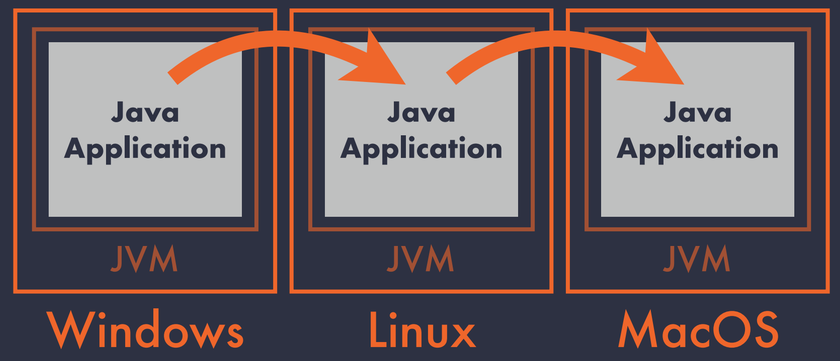
Storage configuration…

Docker

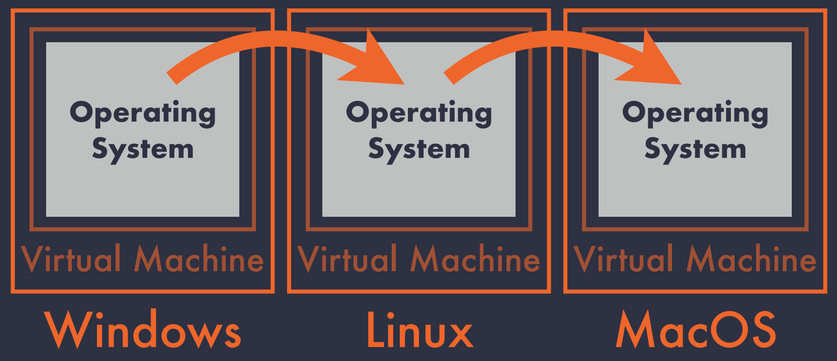
Virtualisation

Virtualisation

Java Virtual Machine

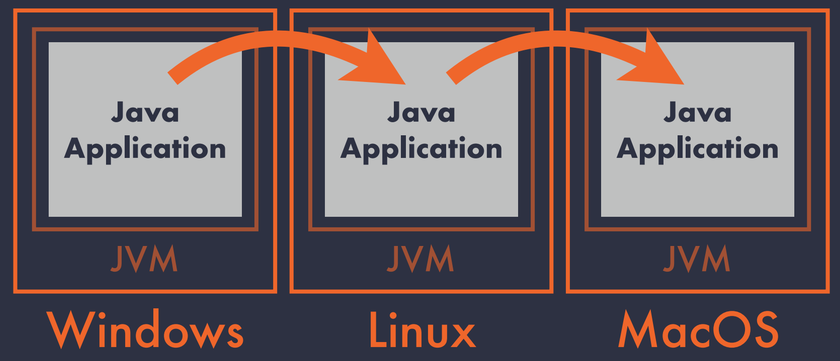


Full Virtualisation

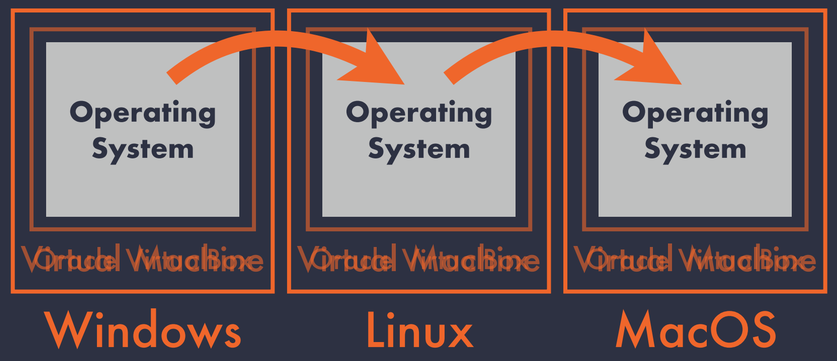


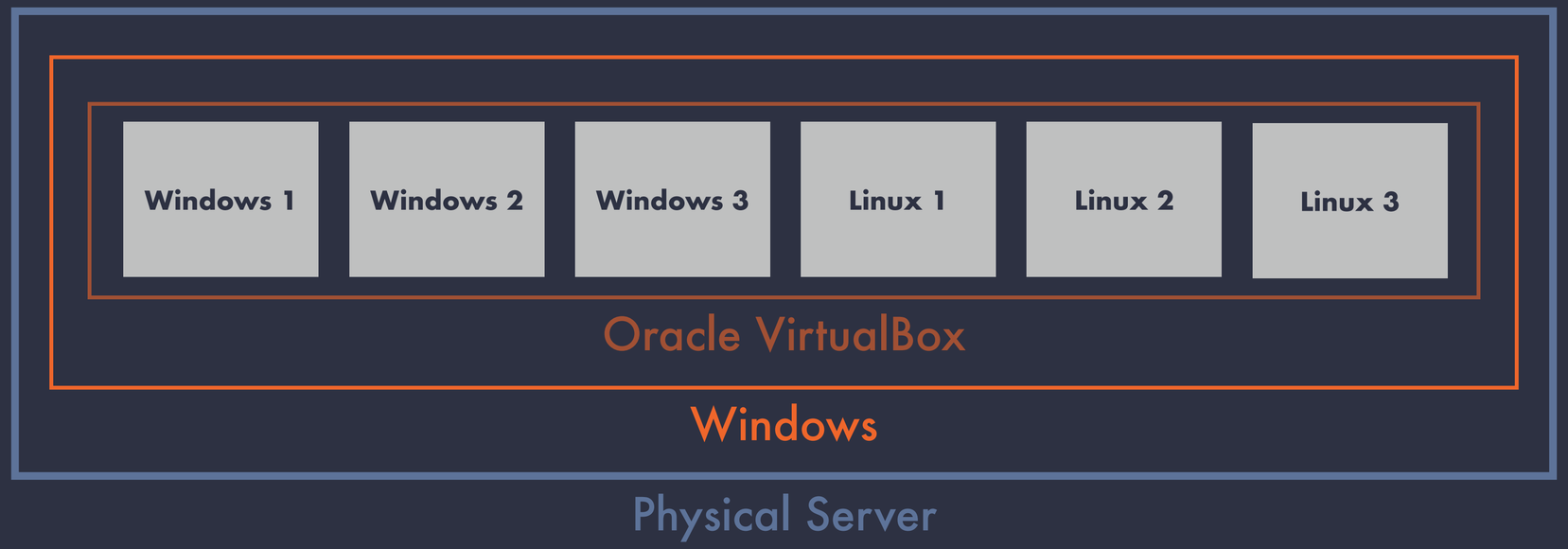
Virtualisation

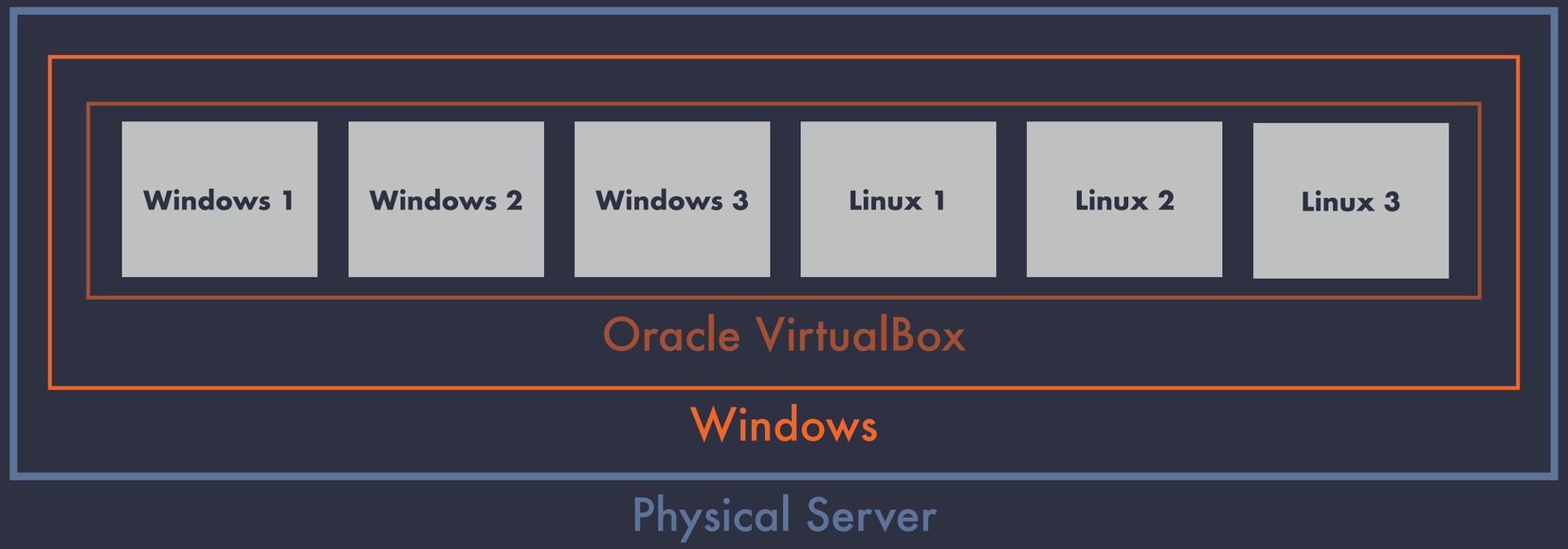
Java Virtual Machine

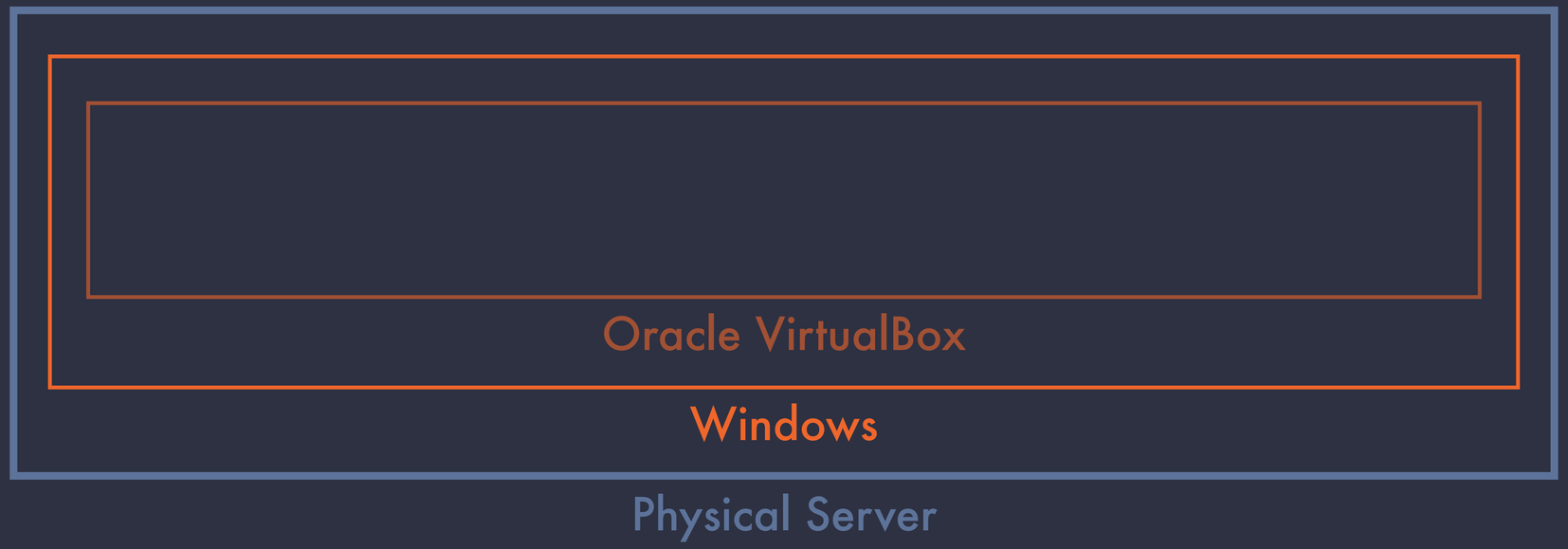


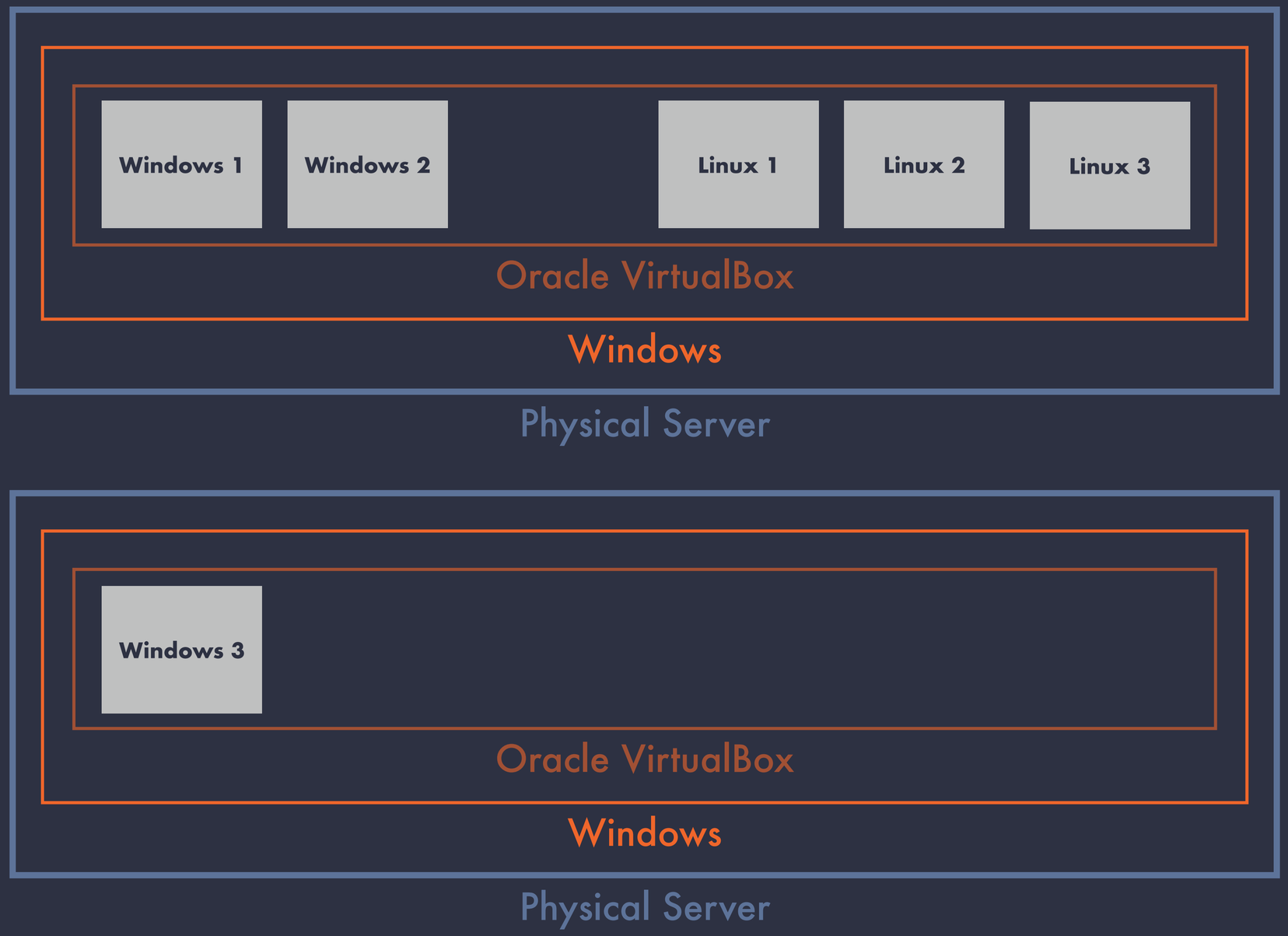
Full Virtualisation

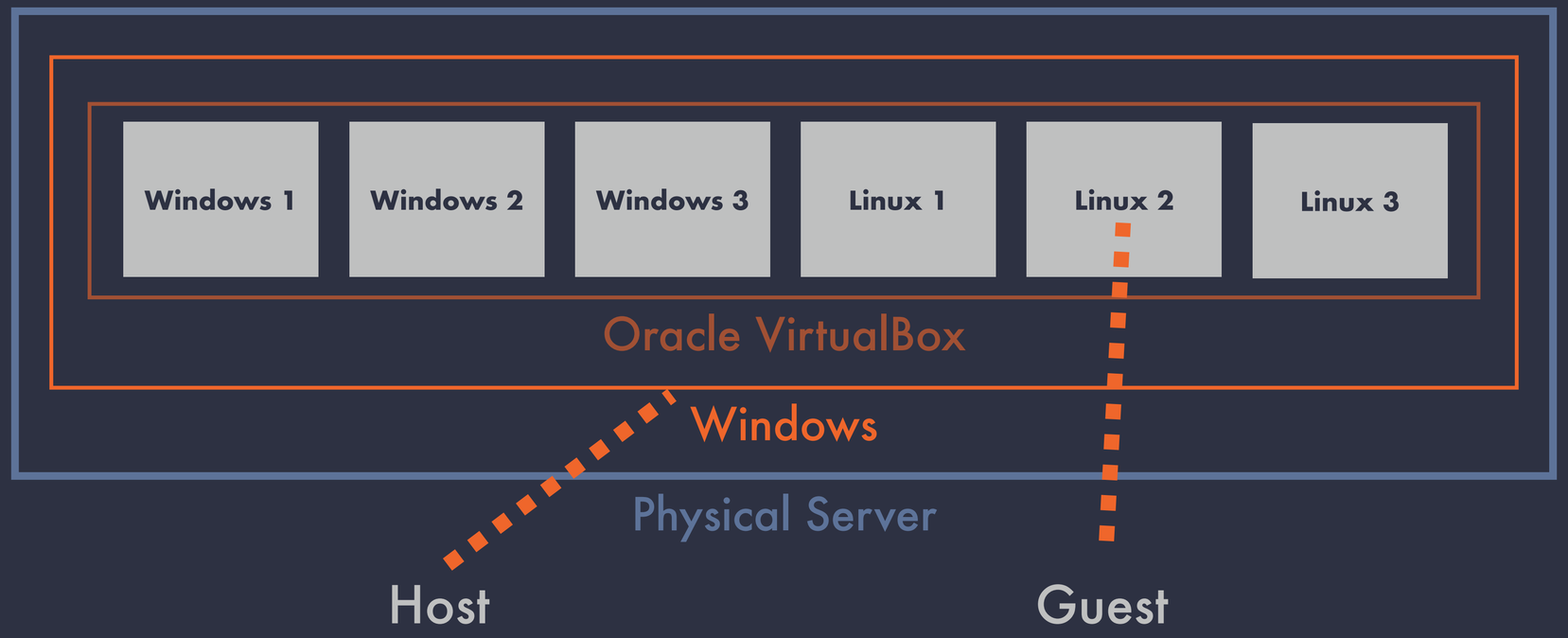



Windows 1
Windows 2
Windows 3
Linux 1
Linux 2
Linux 3
Oracle VirtualBox
Windows
Physical Server

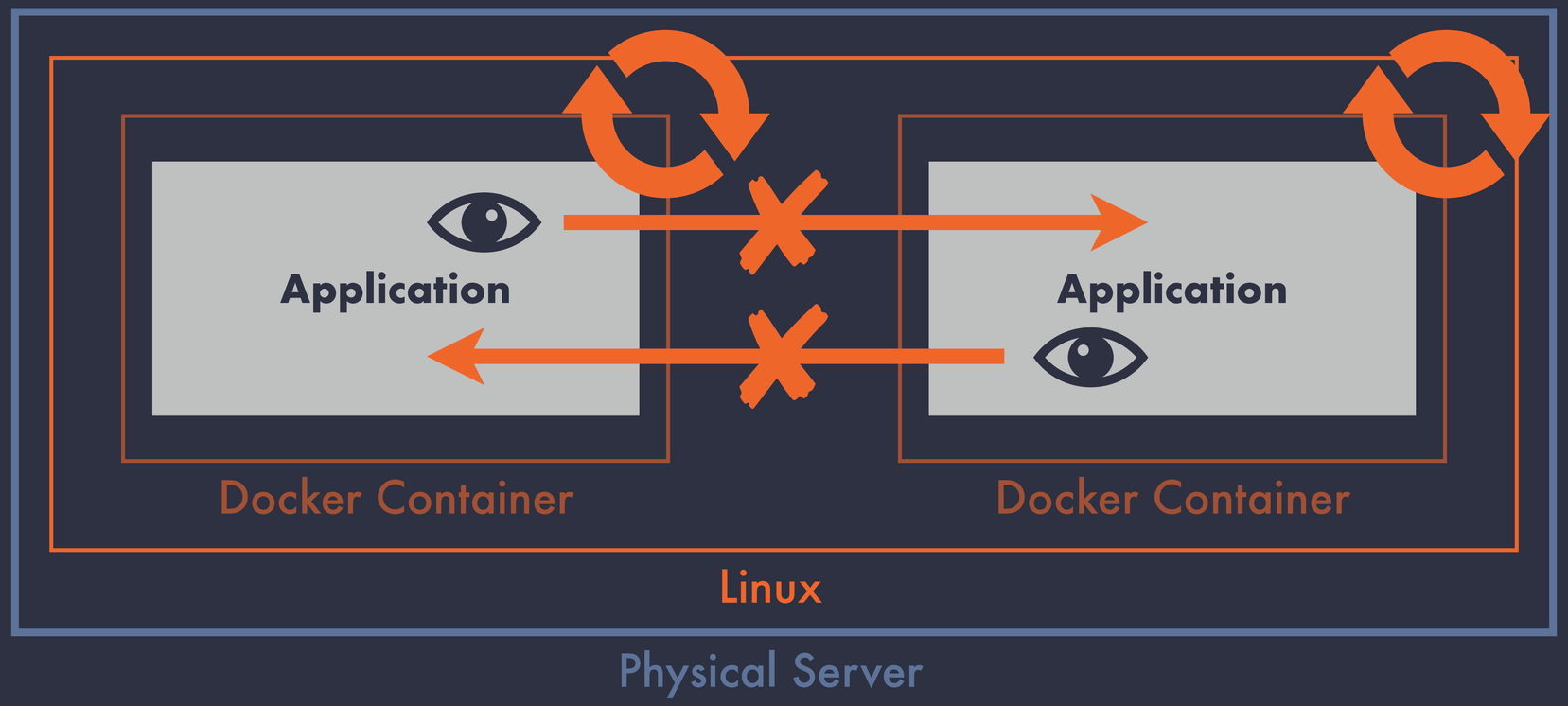

Windows 1
Windows 2
Windows 3
Linux 1
Linux 2
Linux 3
Oracle VirtualBox
Windows
Physical Server


Oracle VirtualBox
Windows
Physical Server

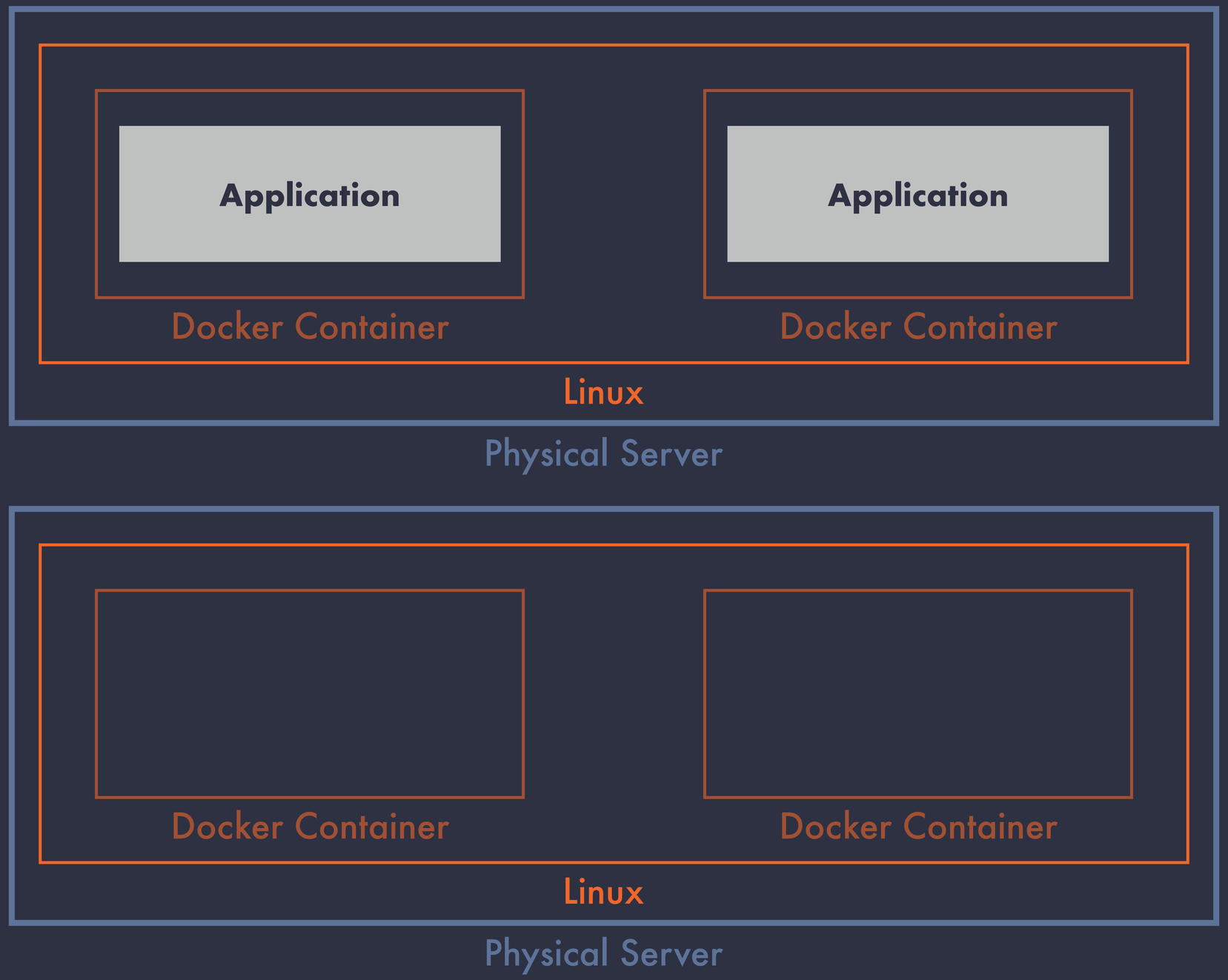

Windows 1
Windows 2
Linux 1
Linux 2
Linux 3
Oracle VirtualBox
Windows
Physical Server
Windows 3
Oracle VirtualBox
Windows
Physical Server

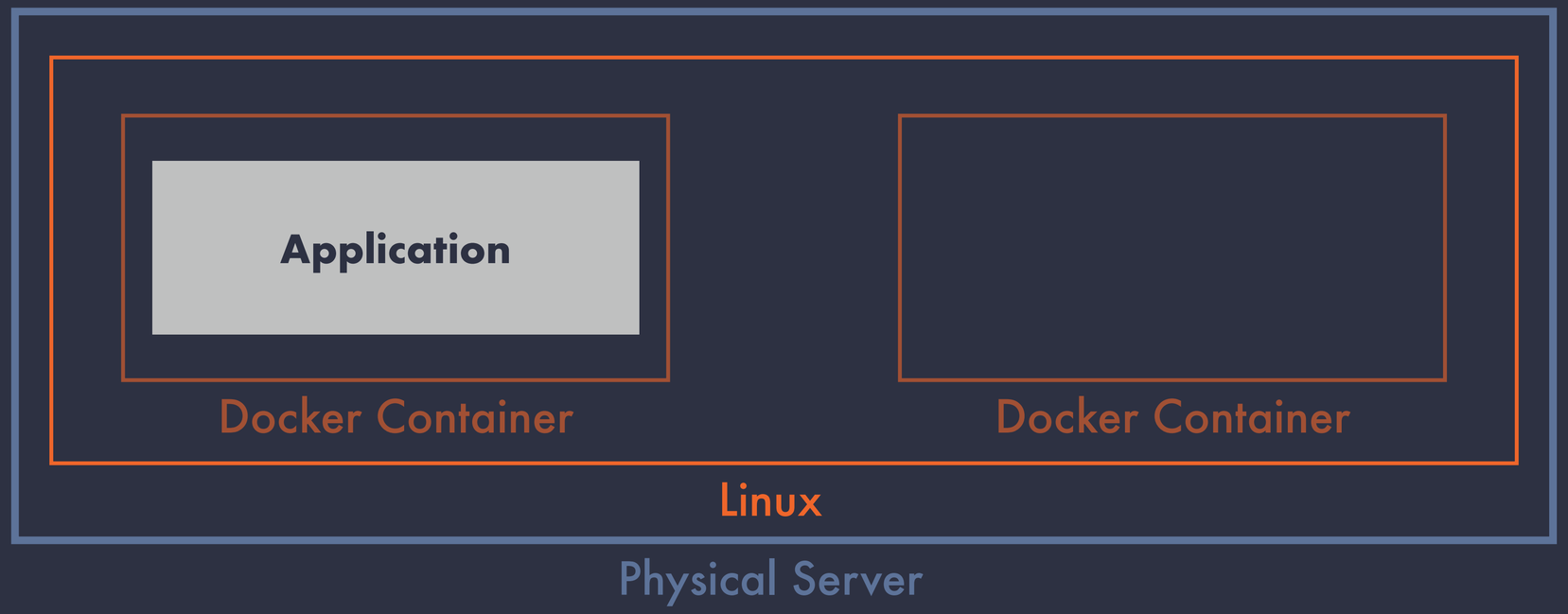


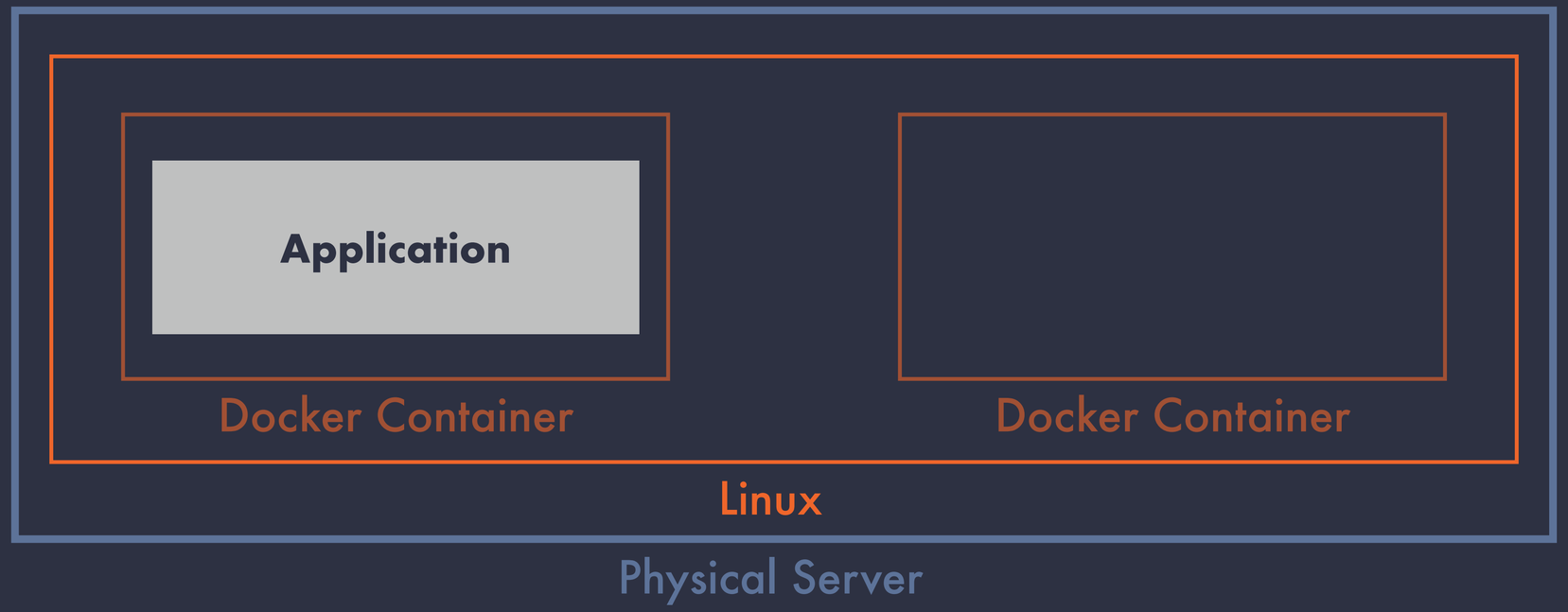
Full operating systems!



Full operating systems!


Application
Application
Docker Container
Docker Container
Linux
Physical Server
Docker Container
Docker Container
Linux
Physical Server


Application
Docker Container
Docker Container
Linux
Physical Server


Application
Docker Container
Docker Container
Linux
Physical Server

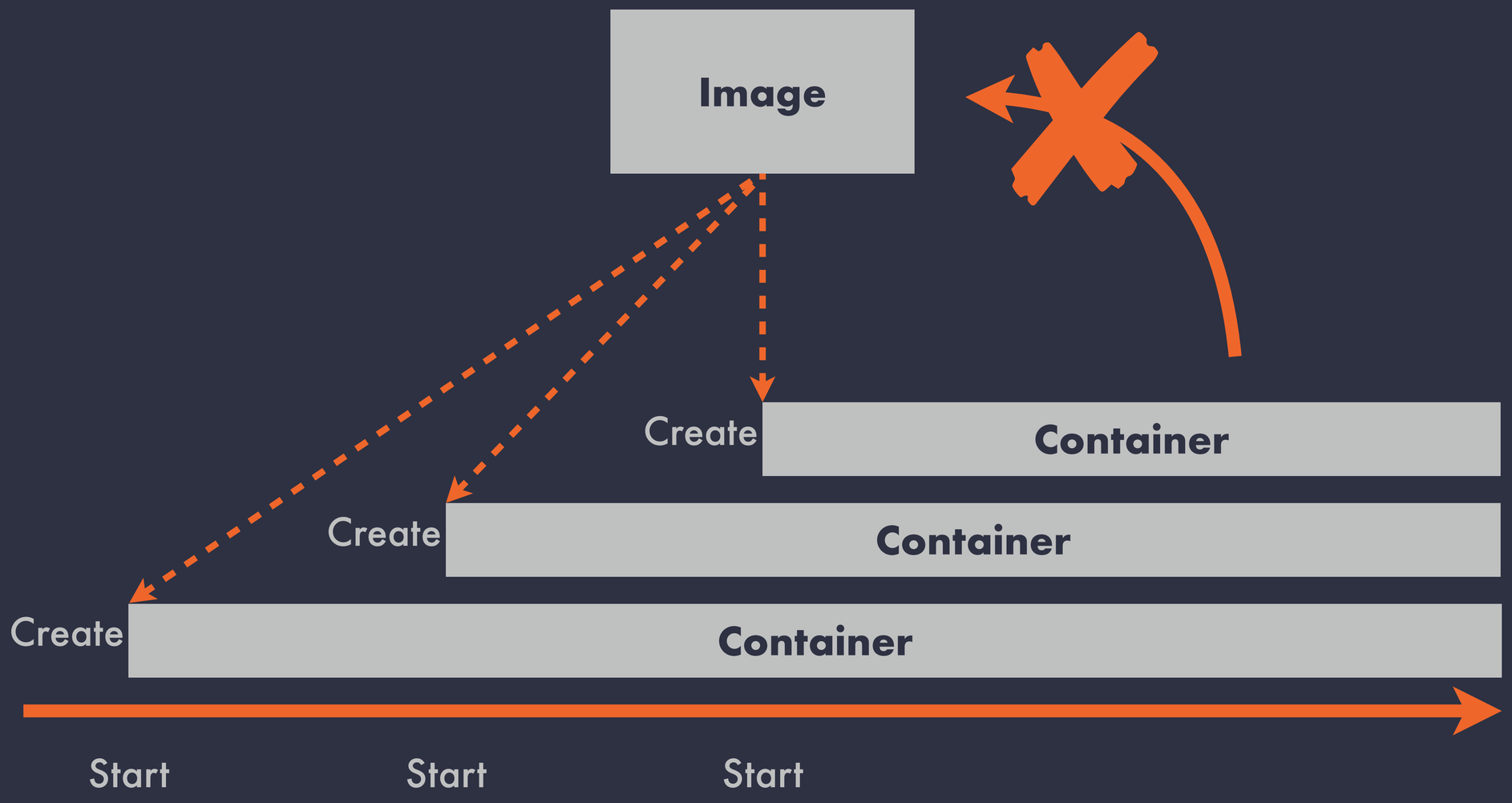
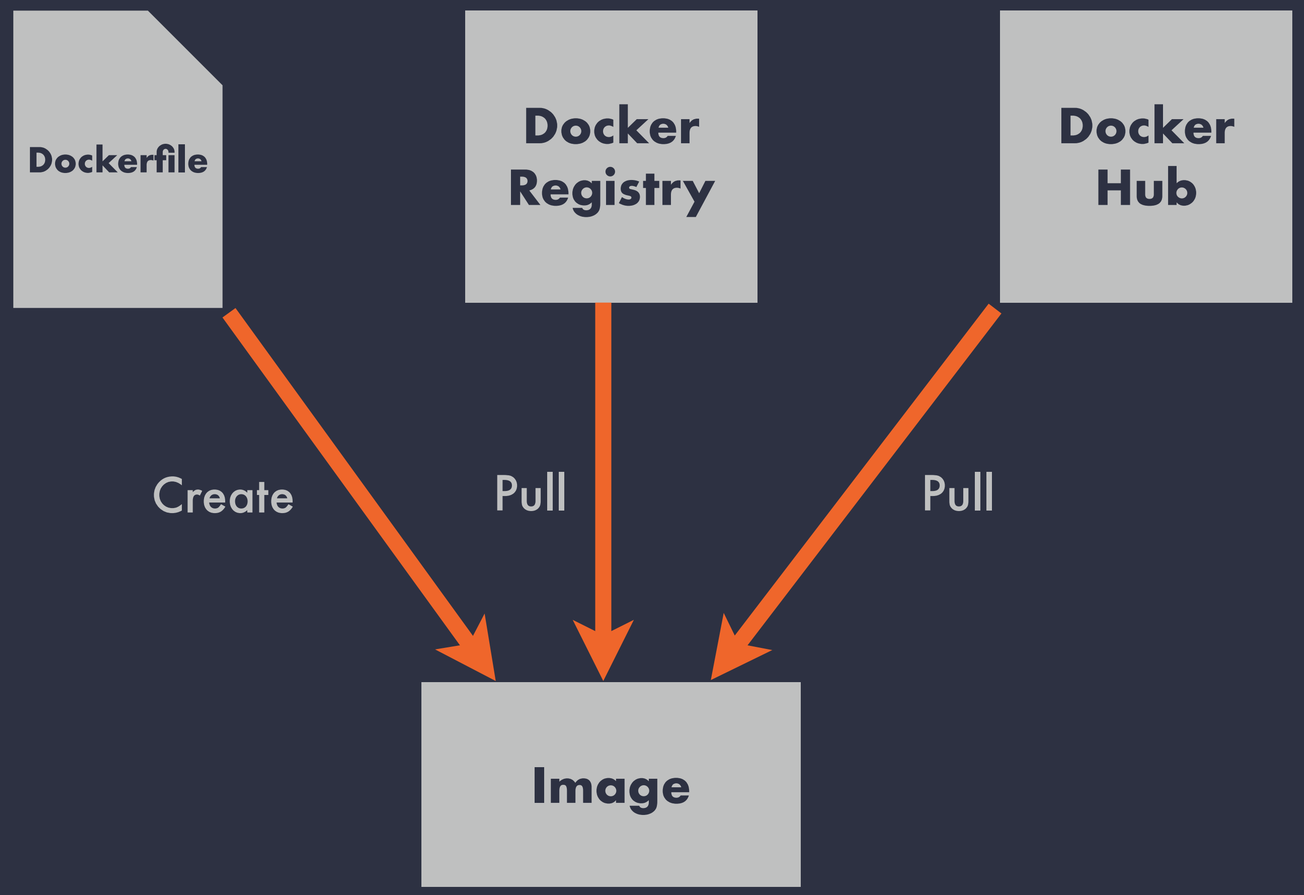


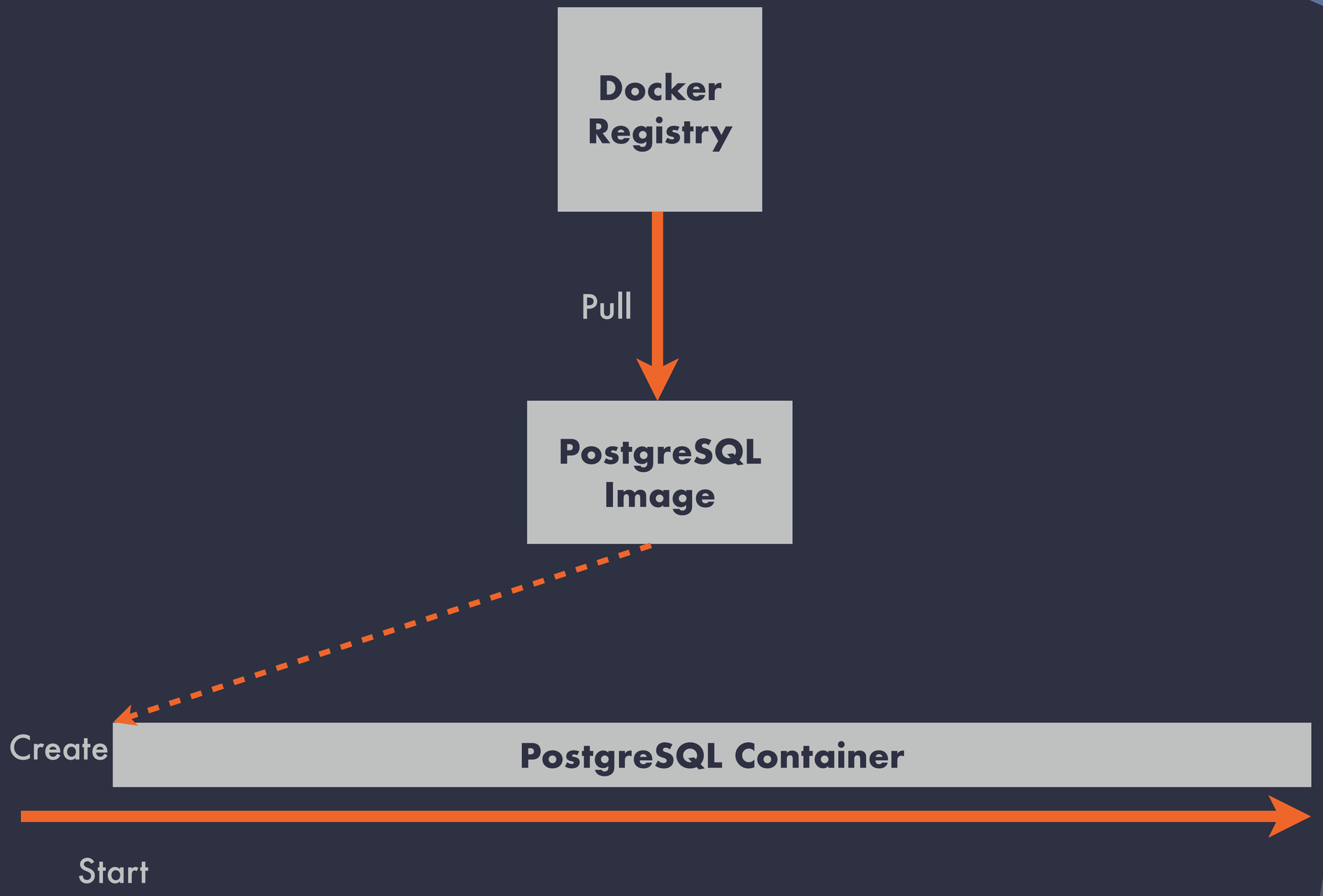
Image
Create
Container
Create
Container
Create
Container
Start
Start
Start

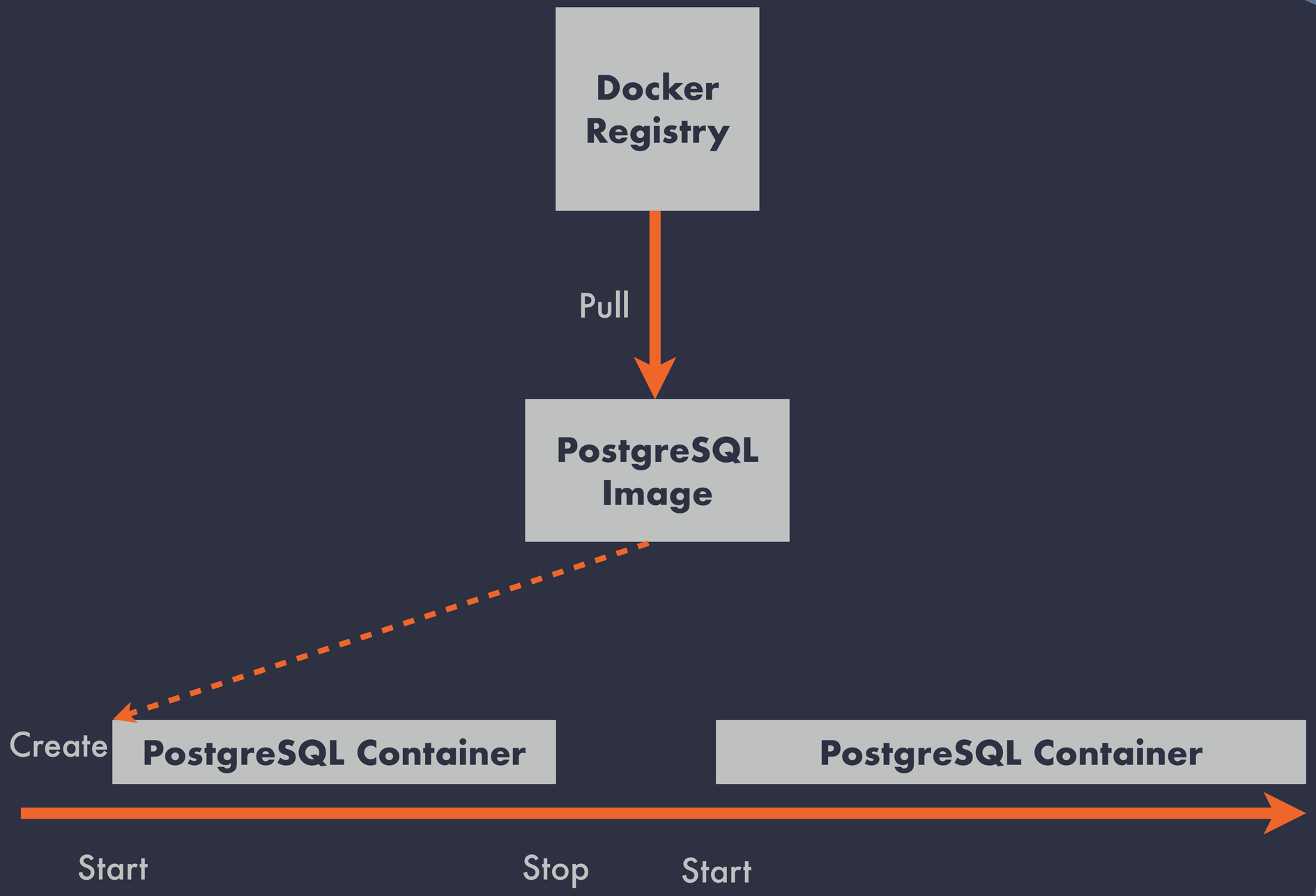

Dockerfile
Docker Registry
Docker Hub
Create
Pull
Pull
Image

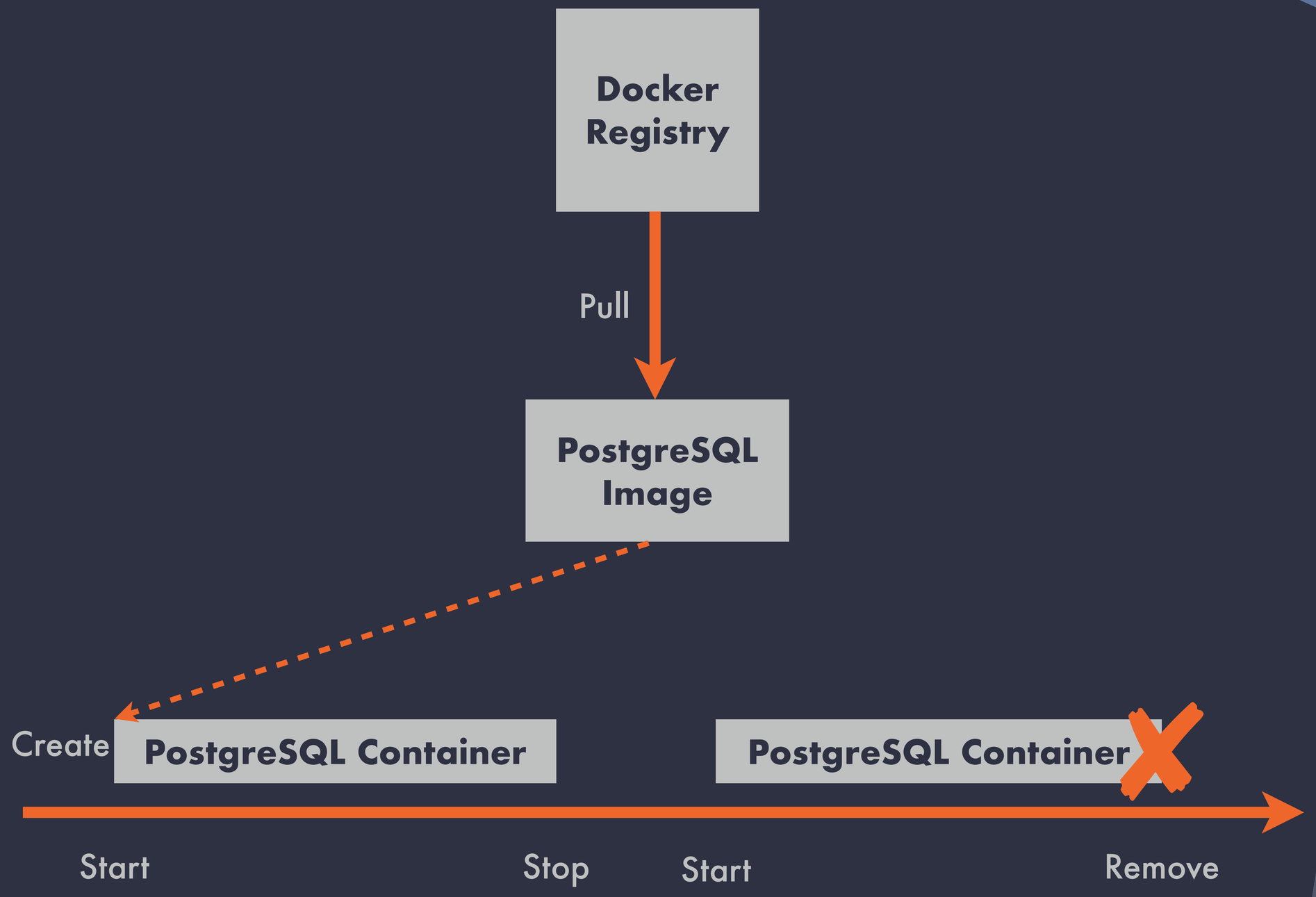
PostgreSQL Docker Image for NrgyInvoicR

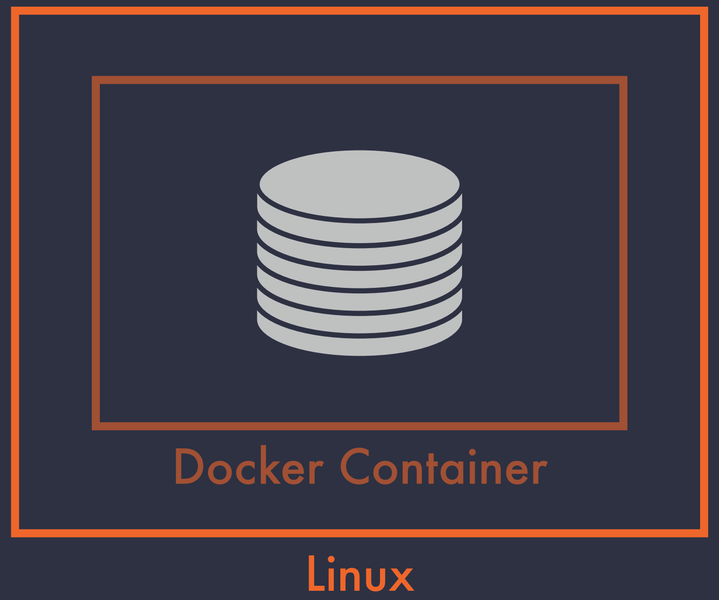
https://hub.docker.com/repository/docker/bkaminnski/postgres-initializable

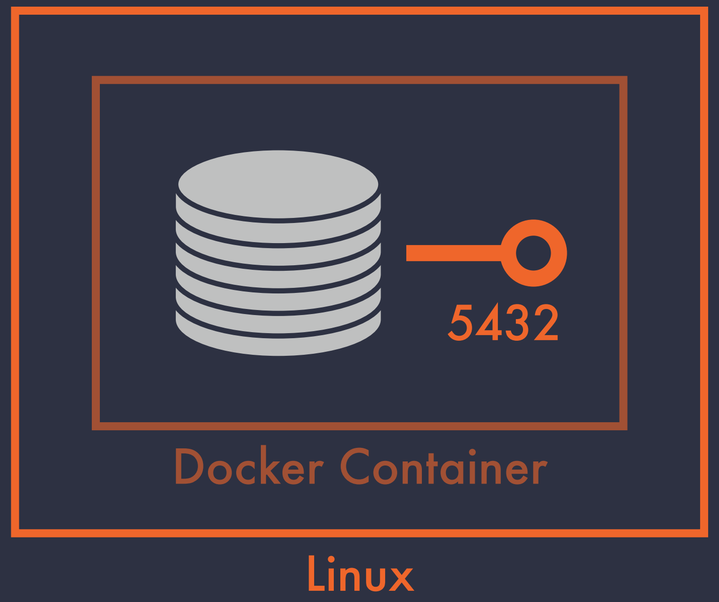
https://github.com/bkaminnski/docker/tree/master/postgres-initializable

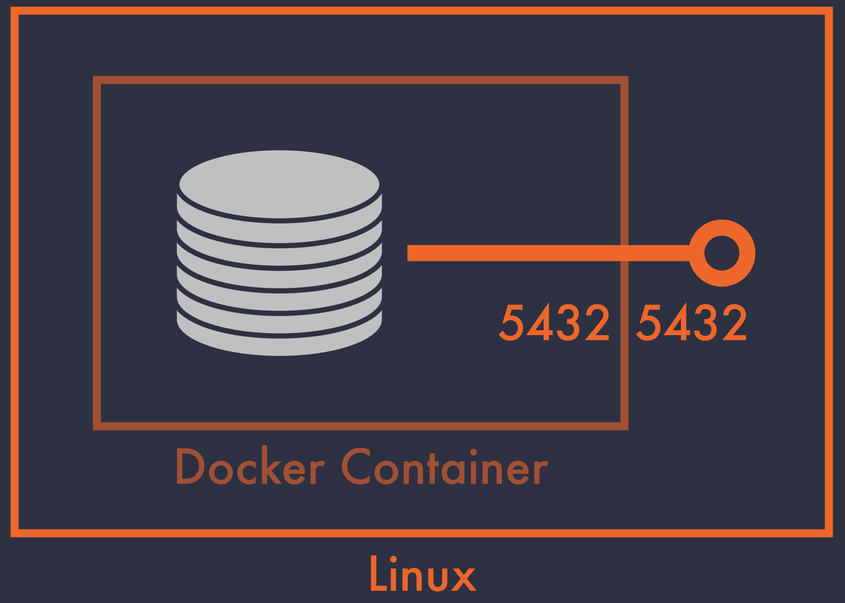

Docker Registry
Pull
PostgreSQL Image
Create
PostgreSQL Container
Start

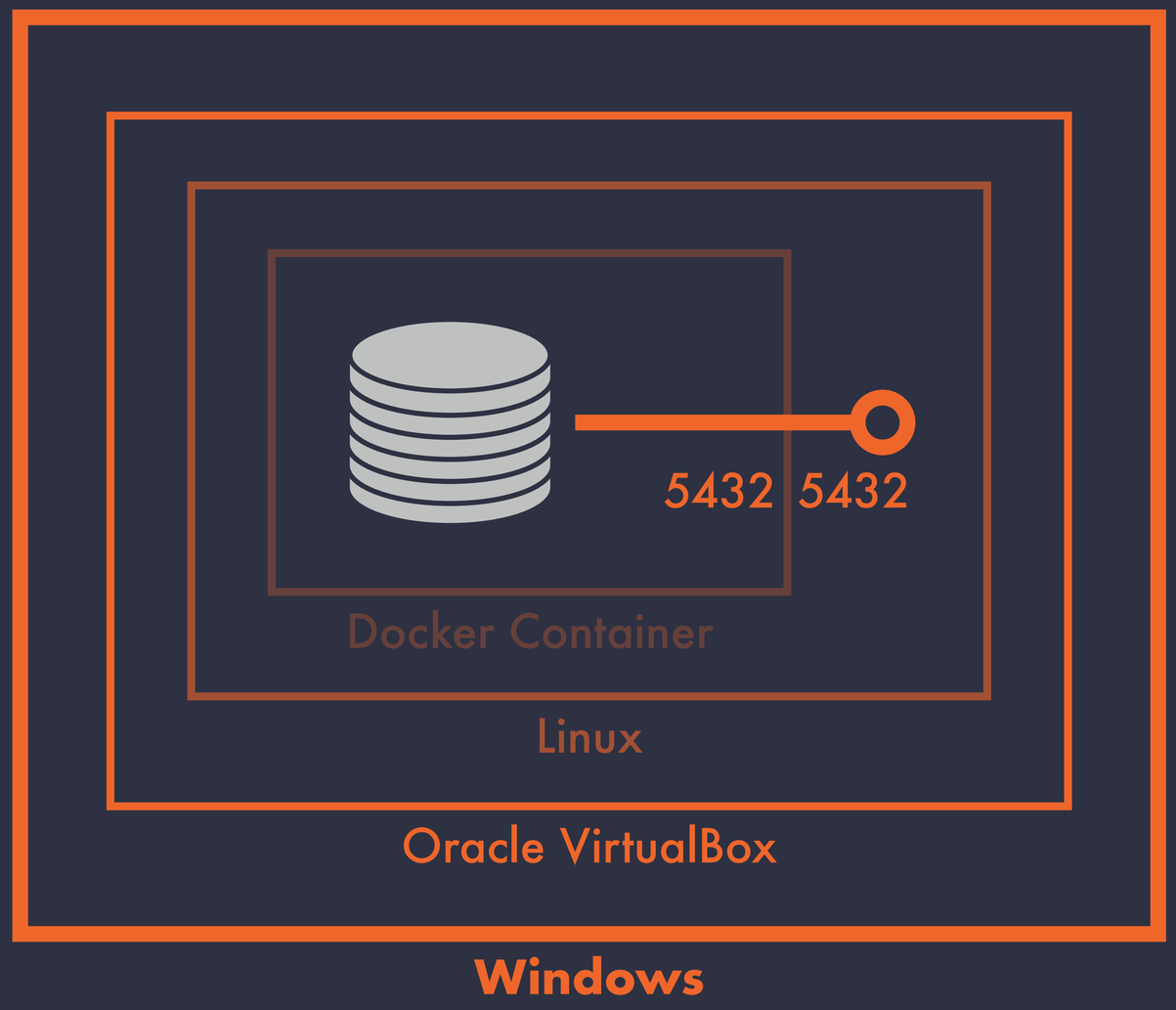

Docker Registry
Pull
PostgreSQL Image
Create
PostgreSQL Container
PostgreSQL Container
Start
Stop
Start

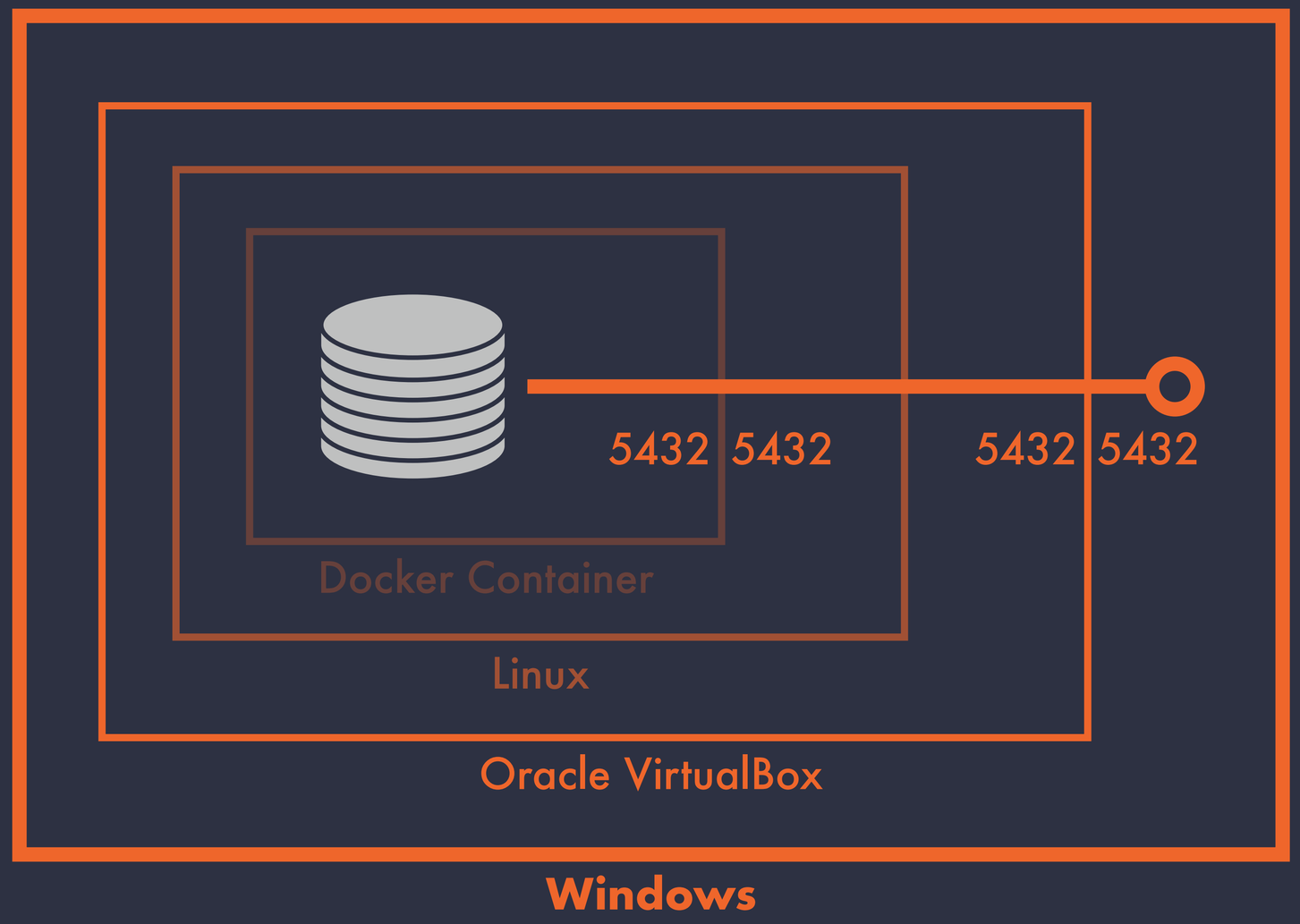

Docker Registry
Pull
PostgreSQL Image
Create
PostgreSQL Container
PostgreSQL Container
Start
Stop
Start
Remove

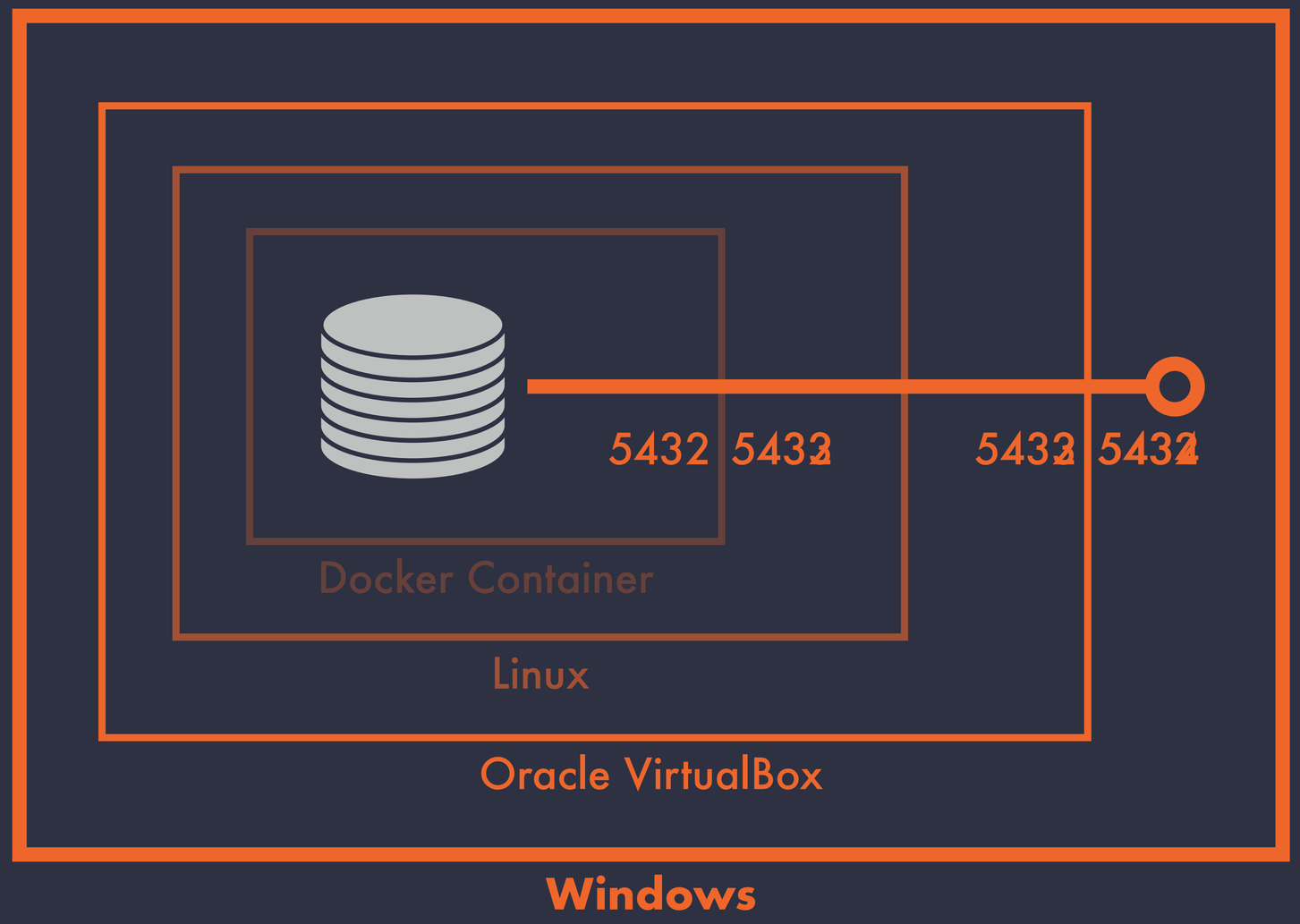

Docker Container
Linux

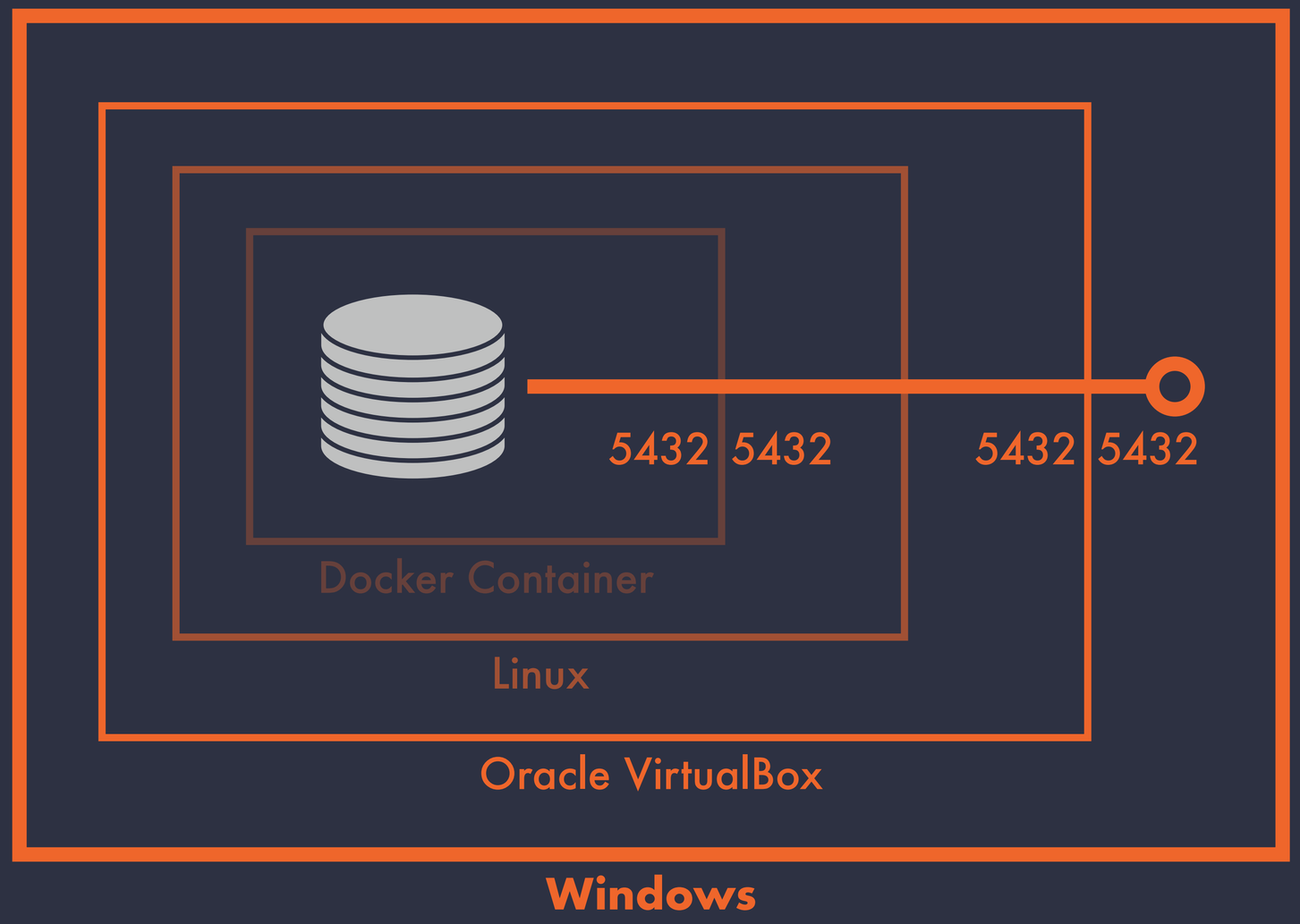

5432
Docker Container
Linux


5432 5432
Docker Container
Linux

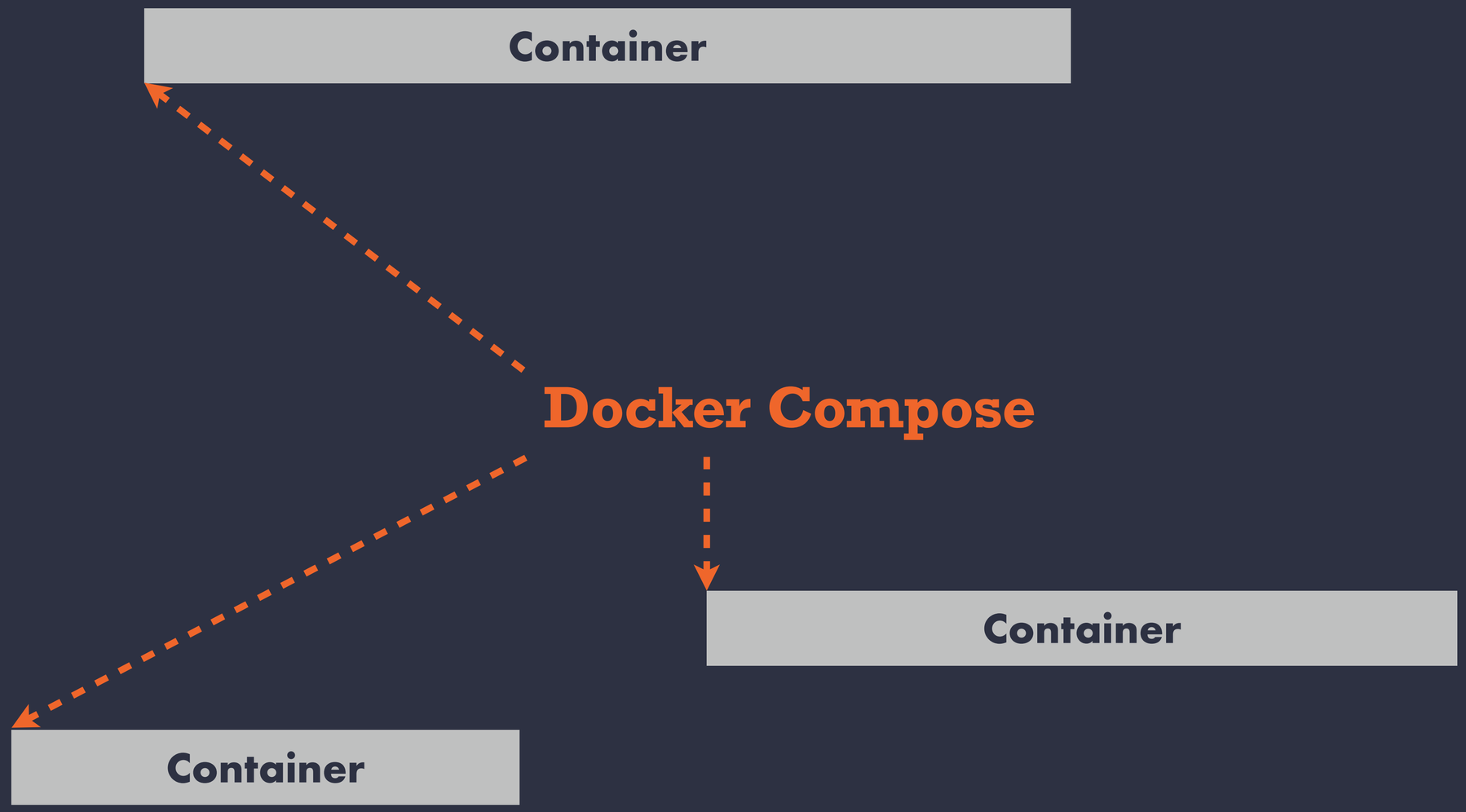

5432 5432
Docker Container
Linux
Oracle VirtualBox
Windows


5432 5432
5432 5432
Docker Container
Linux
Oracle VirtualBox
Windows


5432
5433
5433
5432
Docker Container
Linux
Oracle VirtualBox
Windows


5432 5432
5432 5432
Docker Container
Linux
Oracle VirtualBox
Windows

```
docker build
docker pull
docker run
docker kill
docker stop
docker start
docker rm
...
```


Container
Docker Compose
Container
Container

```
docker pull ...          docker run ...
```

docker-compose up

```yaml
version: '3'

networks:
  nrgyinvoicr_network:

services:

  postgres:
    container_name: postgres
    image: bkaminnski/postgres-initializable
    environment:
    - POSTGRES_PASSWORD=postgres
    - INITIAL_DATABASES=nrgyinvoicr;nrgyinvoicr
    ports:
    - "5432:5432"
    expose:
    - "5432"
    command: postgres
    networks:
    - nrgyinvoicr_network
```

docker-compose.yml

upMachine.psl

Summary

IP address, TCP port

Sample deployment architectures

Docker

Practice a lot!

Docker

Docker → Docker Swarm
Kubernetes

The Pragmatic Programmers
Release It!
Second Edition
Design and Deploy
Production-Ready Software
Michael T. Nygard
Edited by Katharine Dvorak

Module 5

Advanced Software Development

Advanced Software Development

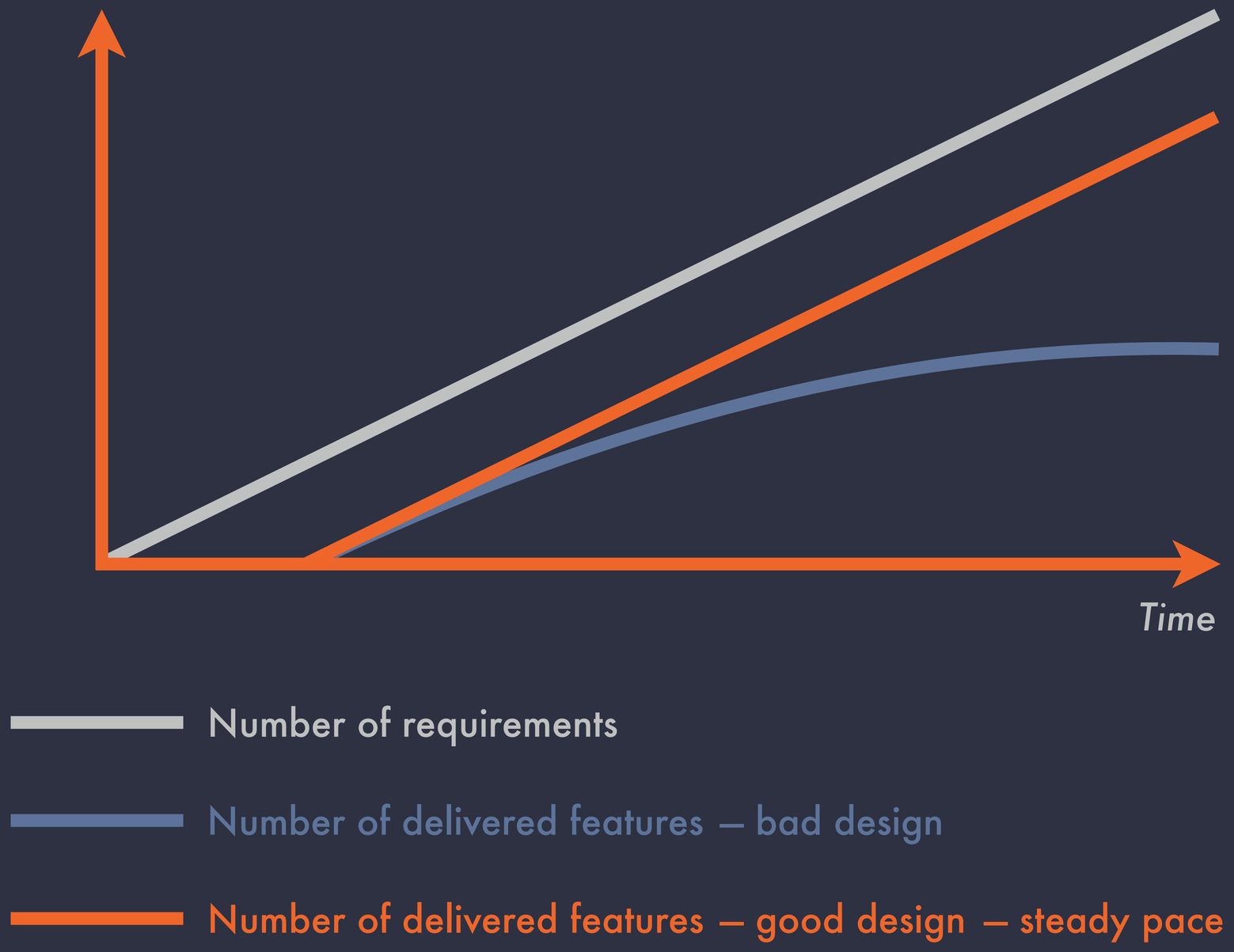
Make it work…

… and think how to make it work better.

Maintainability

Maintainability

Today


Time
Number of requirements
Number of delivered features – bad design
Number of delivered features – good design – steady pace

Guidance

Principles Techniques Practices Disciplines

A
B

A
B

Programming Paradigms

Programming Paradigms

Structural/Procedural

Statements

Conditions

Loops

Procedure call

Object Oriented

Java

TypeScript

Functional

Pure functions

No side effects

From a Bad, Bad Code to the Clean Code

```java
class InvoiceGenerator {

    Invoice generateInvoice(LocalDateTime a, LocalDateTime b, int meter) {
        LinkedList<ReadingValue> rvs = new LinkedList<>();
        LocalDateTime tmp = a;
        while (tmp.isBefore(b))
        {
        BigDecimal v =
                meter % 2 == 0 ? new BigDecimal("0.5")
                        :
                        new BigDecimal("1.0");

        ReadingValue tmp2 = new ReadingValue(tmp, v);
        rvs.add(tmp2);
        tmp = tmp.plusHours(1);
        }
        BigDecimal v = ZERO;
        BigDecimal value2 = ZERO;
        for(ReadingValue r:rvs){
        if(r.getTimestamp().getHour() >= 8
        && r.getTimestamp().getHour() < 23){
        v = v.add(r.getValue());
        }else{
        value2 = value2.add(r.getValue());
        }
        }
        ArrayList<InvoiceLine> ls = new ArrayList<>();
        InvoiceLine line = new InvoiceLine("Day Rate", new BigDecimal("0.19288"),
                v, v.multiply(new BigDecimal("0.19288")));
        InvoiceLine l2 = new InvoiceLine("Night Rate", new BigDecimal("0.17531"),
                value2, value2.multiply(new BigDecimal("0.17531")));
        ls.add(line);
        ls.add(l2);
        BigDecimal x = ls.stream().map(InvoiceLine::getTotalAmount).reduce(ZERO, BigDecimal::add);
        return new Invoice(x, ls);
    }
}
```

```java
class InvoiceGenerator {

    Invoice generateInvoice(LocalDateTime a, LocalDateTime b, int meter) {
        LinkedList<ReadingValue> rvs = new LinkedList<>();
        LocalDateTime tmp = a;
        while (tmp.isBefore(b)) {
            BigDecimal v = meter % 2 == 0 ? new BigDecimal("0.5") : new BigDecimal("1.0");
            ReadingValue tmp2 = new ReadingValue(tmp, v);
            rvs.add(tmp2);
            tmp = tmp.plusHours(1);
        }
        BigDecimal v = ZERO;
        BigDecimal value2 = ZERO;
        for (ReadingValue r : rvs) {
            if (r.getTimestamp().getHour() >= 8
                    && r.getTimestamp().getHour() < 23) {
                v = v.add(r.getValue());
            } else {
                value2 = value2.add(r.getValue());
            }
        }
        ArrayList<InvoiceLine> ls = new ArrayList<>();
        InvoiceLine line = new InvoiceLine("Day Rate", new BigDecimal("0.19288"),
                v, v.multiply(new BigDecimal("0.19288")));
        InvoiceLine l2 = new InvoiceLine("Night Rate", new BigDecimal("0.17531"),
                value2, value2.multiply(new BigDecimal("0.17531")));
        ls.add(line);
        ls.add(l2);
        BigDecimal x = ls.stream()
                .map(InvoiceLine::getTotalAmount)
                .reduce(ZERO, BigDecimal::add);
        return new Invoice(x, ls);
    }
}
```

https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

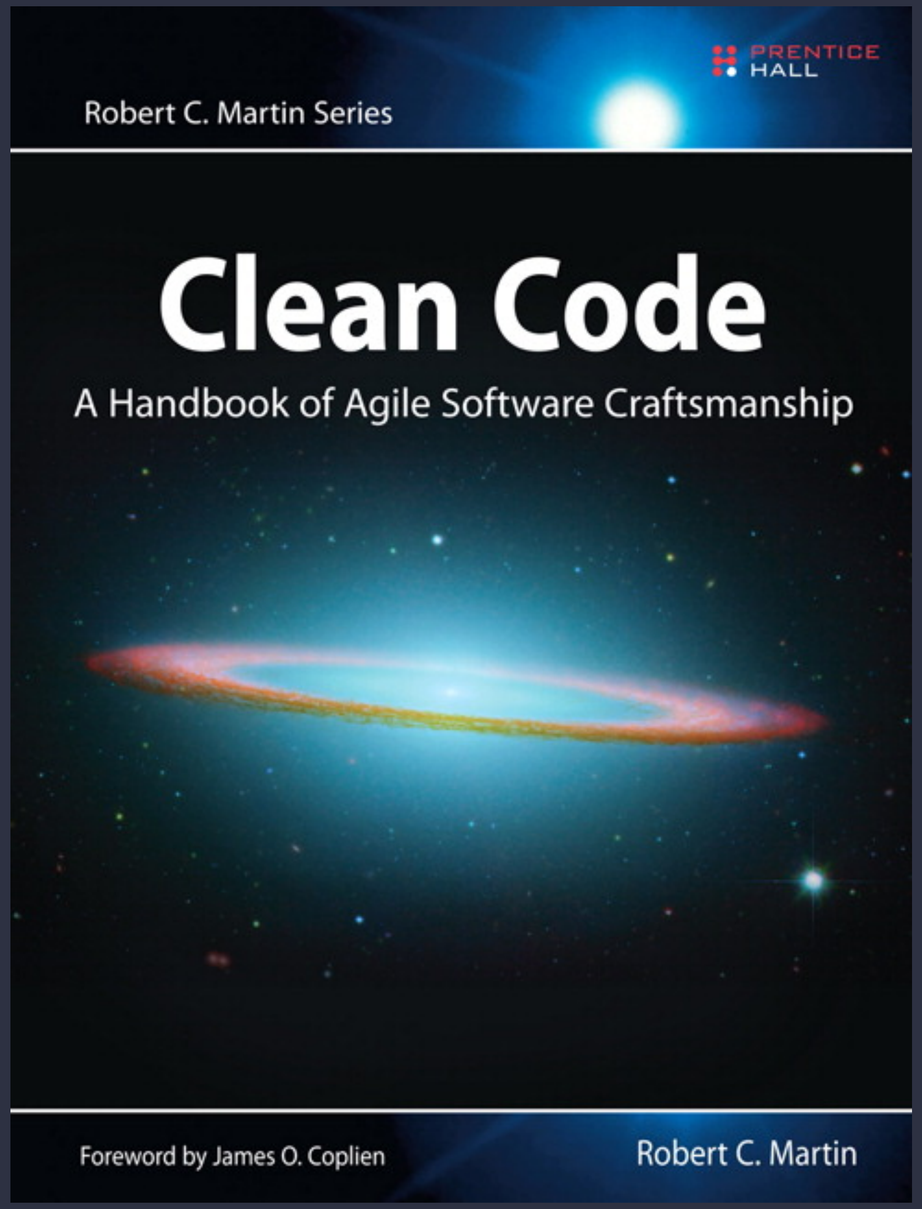
```java
class InvoiceGenerator {

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        LinkedList<ReadingValue> readingValues = new LinkedList<>();
        LocalDateTime timestamp = sinceClosed;
        while (timestamp.isBefore(untilOpen)) {
            BigDecimal value = meterId % 2 == 0 ? new BigDecimal("0.5") : new BigDecimal("1.0");
            ReadingValue readingValue = new ReadingValue(timestamp, value);
            readingValues.add(readingValue);
            timestamp = timestamp.plusHours(1);
        }
        BigDecimal dayUsage = ZERO;
        BigDecimal nightUsage = ZERO;
        for (ReadingValue readingValue : readingValues) {
            if (readingValue.getTimestamp().getHour() >= 8
                    && readingValue.getTimestamp().getHour() < 23) {
                dayUsage = dayUsage.add(readingValue.getValue());
            } else {
                nightUsage = nightUsage.add(readingValue.getValue());
            }
        }
        ArrayList<InvoiceLine> invoiceLines = new ArrayList<>();
        InvoiceLine dayInvoiceLine = new InvoiceLine("Day Rate", new BigDecimal("0.19288"),
                dayUsage, dayUsage.multiply(new BigDecimal("0.19288")));
        InvoiceLine nightInvoiceLine = new InvoiceLine("Night Rate", new BigDecimal("0.17531"),
                nightUsage, nightUsage.multiply(new BigDecimal("0.17531")));
        invoiceLines.add(dayInvoiceLine);
        invoiceLines.add(nightInvoiceLine);
        BigDecimal totalAmount = invoiceLines.stream()
                .map(InvoiceLine::getTotalAmount)
                .reduce(ZERO, BigDecimal::add);
        return new Invoice(totalAmount, invoiceLines);
    }
}
```

```java
class InvoiceGenerator {
    private static final int DAY_START = 8;
    private static final int DAY_END = 23;
    private static final BigDecimal DAY_RATE = new BigDecimal("0.19288");
    private static final BigDecimal NIGHT_RATE = new BigDecimal("0.17531");

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        LinkedList<ReadingValue> readingValues = new LinkedList<>();
        LocalDateTime timestamp = sinceClosed;
        while (timestamp.isBefore(untilOpen)) {
            BigDecimal value = meterId % 2 == 0 ? new BigDecimal("0.5") : new BigDecimal("1.0");
            ReadingValue readingValue = new ReadingValue(timestamp, value);
            readingValues.add(readingValue);
            timestamp = timestamp.plusHours(1);
        }
        BigDecimal dayUsage = ZERO;
        BigDecimal nightUsage = ZERO;
        for (ReadingValue readingValue : readingValues) {
            if (readingValue.getTimestamp().getHour() >= DAY_START
                    && readingValue.getTimestamp().getHour() < DAY_END) {
                dayUsage = dayUsage.add(readingValue.getValue());
            } else {
                nightUsage = nightUsage.add(readingValue.getValue());
            }
        }
        ArrayList<InvoiceLine> invoiceLines = new ArrayList<>();
        InvoiceLine dayInvoiceLine = new InvoiceLine("Day Rate", DAY_RATE,
                dayUsage, dayUsage.multiply(DAY_RATE));
        InvoiceLine nightInvoiceLine = new InvoiceLine("Night Rate", NIGHT_RATE,
                nightUsage, nightUsage.multiply(NIGHT_RATE));
        invoiceLines.add(dayInvoiceLine);
        invoiceLines.add(nightInvoiceLine);
        BigDecimal totalAmount = invoiceLines.stream()
                .map(InvoiceLine::getTotalAmount)
                .reduce(ZERO, BigDecimal::add);
        return new Invoice(totalAmount, invoiceLines);
    }
}
```

```java
class InvoiceGenerator {
    private static final int DAY_START = 8;
    private static final int DAY_END = 23;
    private static final BigDecimal DAY_RATE = new BigDecimal("0.19288");
    private static final BigDecimal NIGHT_RATE = new BigDecimal("0.17531");

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        LinkedList<ReadingValue> readingValues = new LinkedList<>();
        LocalDateTime timestamp = sinceClosed;
        while (timestamp.isBefore(untilOpen)) {
            BigDecimal value = meterId % 2 == 0 ? new BigDecimal("0.5") : new BigDecimal("1.0");
            ReadingValue readingValue = new ReadingValue(timestamp, value);
            readingValues.add(readingValue);
            timestamp = timestamp.plusHours(1);
        }
        BigDecimal dayUsage = ZERO;
        BigDecimal nightUsage = ZERO;
        for (ReadingValue readingValue : readingValues) {
            if (readingValue.getTimestamp().getHour() >= DAY_START
                    && readingValue.getTimestamp().getHour() < DAY_END) {
                dayUsage = dayUsage.add(readingValue.getValue());
            } else {
                nightUsage = nightUsage.add(readingValue.getValue());
            }
        }
        ArrayList<InvoiceLine> invoiceLines = new ArrayList<>();
        InvoiceLine dayInvoiceLine = new InvoiceLine("Day Rate", DAY_RATE,
                dayUsage, dayUsage.multiply(DAY_RATE));
        InvoiceLine nightInvoiceLine = new InvoiceLine("Night Rate", NIGHT_RATE,
                nightUsage, nightUsage.multiply(NIGHT_RATE));
        invoiceLines.add(dayInvoiceLine);
        invoiceLines.add(nightInvoiceLine);
        return new Invoice(invoiceLines);
    }
}
```

```
class InvoiceGenerator {

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        LinkedList<ReadingValue> readingValues = new LinkedList<>();
        LocalDateTime timestamp = sinceClosed;
        while (timestamp.isBefore(untilOpen)) {
            BigDecimal value = meterId % 2 == 0 ? new BigDecimal("0.5") : new BigDecimal("1.0");
            ReadingValue readingValue = new ReadingValue(timestamp, value);
            readingValues.add(readingValue);
            timestamp = timestamp.plusHours(1);
        }
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

```
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

Abstraction

Abstraction is the ability to engage with a concept while safely ignoring some of its details (...).

Code Complete, 2nd Edition
Steve McConnell

```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
```

```java
class NightPlan {
    private static final int DAY_START = 8;
    private static final int DAY_END = 23;
    private static final BigDecimal DAY_RATE = new BigDecimal("0.19288");
    private static final BigDecimal NIGHT_RATE = new BigDecimal("0.17531");

    private BigDecimal dayUsage = ZERO;
    private BigDecimal nightUsage = ZERO;

    void addReadingValues(List<ReadingValue> readingValues) {
        readingValues.forEach(this::addReadingValue);
    }

    private void addReadingValue(ReadingValue readingValue) {
        if (readingValue.getTimestamp().getHour() >= DAY_START
                && readingValue.getTimestamp().getHour() < DAY_END) {
            dayUsage = dayUsage.add(readingValue.getValue());
        } else {
            nightUsage = nightUsage.add(readingValue.getValue());
        }
    }

    List<InvoiceLine> toInvoiceLines() {
        ArrayList<InvoiceLine> invoiceLines = new ArrayList<>();
        InvoiceLine dayInvoiceLine = new InvoiceLine("Day Rate", DAY_RATE,
                dayUsage, dayUsage.multiply(DAY_RATE));
        InvoiceLine nightInvoiceLine = new InvoiceLine("Night Rate", NIGHT_RATE,
                nightUsage, nightUsage.multiply(NIGHT_RATE));
        invoiceLines.add(dayInvoiceLine);
        invoiceLines.add(nightInvoiceLine);
        return invoiceLines;
    }
}
```

Encapsulation

Encapsulation picks up where abstraction leaves off.

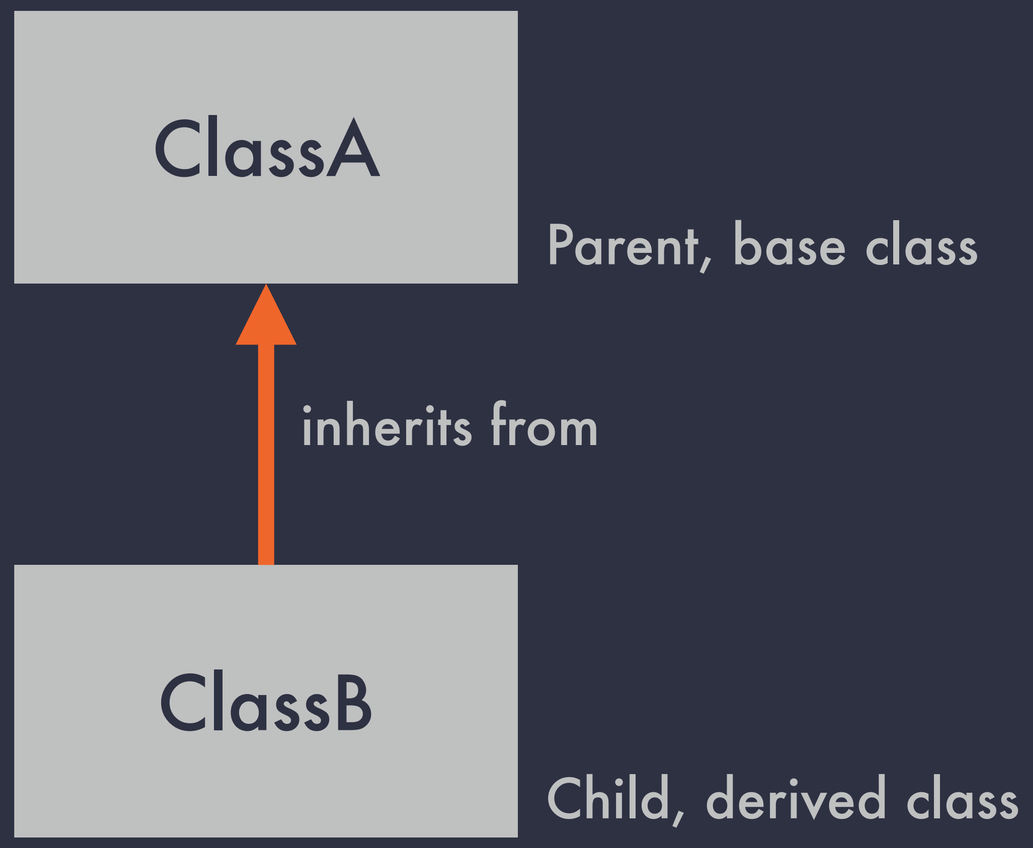
Code Complete, 2nd Edition
Steve McConnell

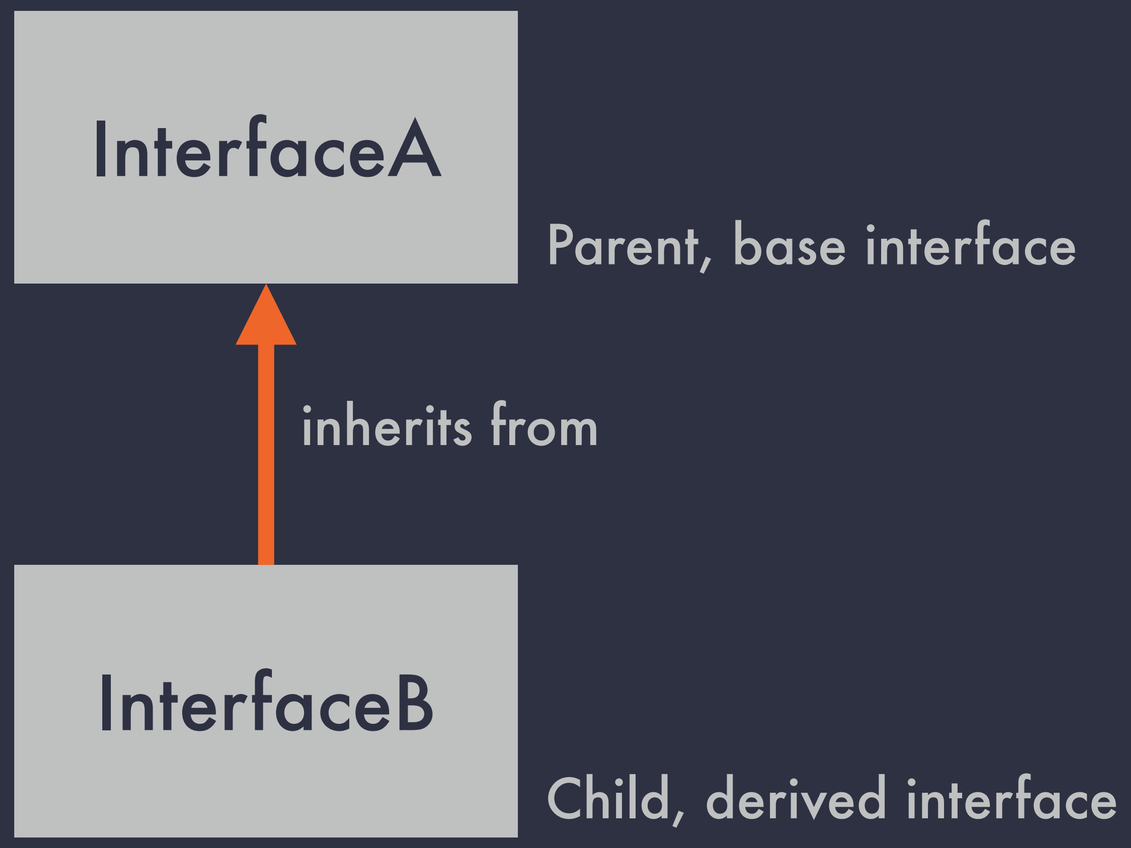
```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
```

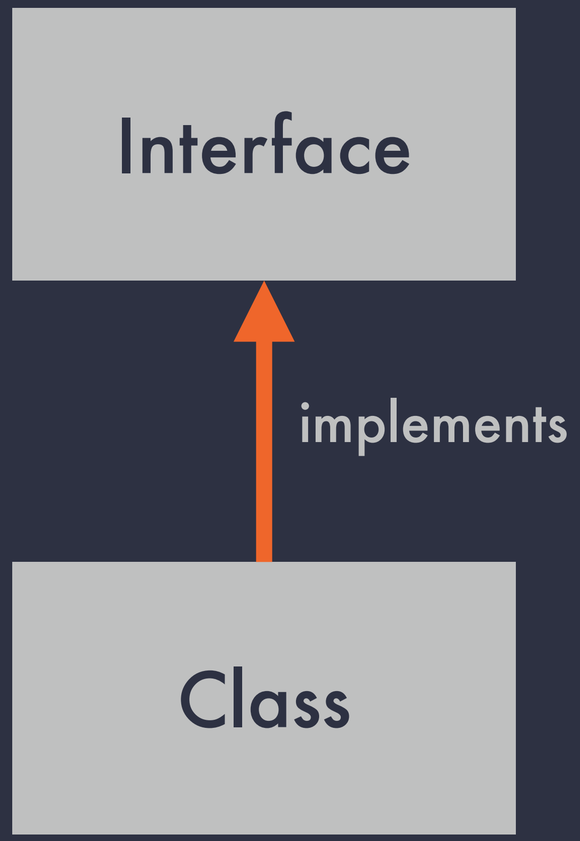
```java
class NightPlan {
    private static final int DAY_START = 8;
    private static final int DAY_END = 23;
    private static final BigDecimal DAY_RATE = new BigDecimal("0.19288");
    private static final BigDecimal NIGHT_RATE = new BigDecimal("0.17531");

    private BigDecimal dayUsage = ZERO;
    private BigDecimal nightUsage = ZERO;

    void addReadingValues(List<ReadingValue> readingValues) {
        readingValues.forEach(this::addReadingValue);
    }

    private void addReadingValue(ReadingValue readingValue) {
        if (readingValue.getTimestamp().getHour() >= DAY_START
                && readingValue.getTimestamp().getHour() < DAY_END) {
            dayUsage = dayUsage.add(readingValue.getValue());
        } else {
            nightUsage = nightUsage.add(readingValue.getValue());
        }
    }

    List<InvoiceLine> toInvoiceLines() {
        ArrayList<InvoiceLine> invoiceLines = new ArrayList<>();
        InvoiceLine dayInvoiceLine = new InvoiceLine("Day Rate", DAY_RATE,
                dayUsage, dayUsage.multiply(DAY_RATE));
        InvoiceLine nightInvoiceLine = new InvoiceLine("Night Rate", NIGHT_RATE,
                nightUsage, nightUsage.multiply(NIGHT_RATE));
        invoiceLines.add(dayInvoiceLine);
        invoiceLines.add(nightInvoiceLine);
        return invoiceLines;
    }
}
```

Inheritance

Polymorphism


ClassA
Parent, base class
inherits from
ClassB
Child, derived class

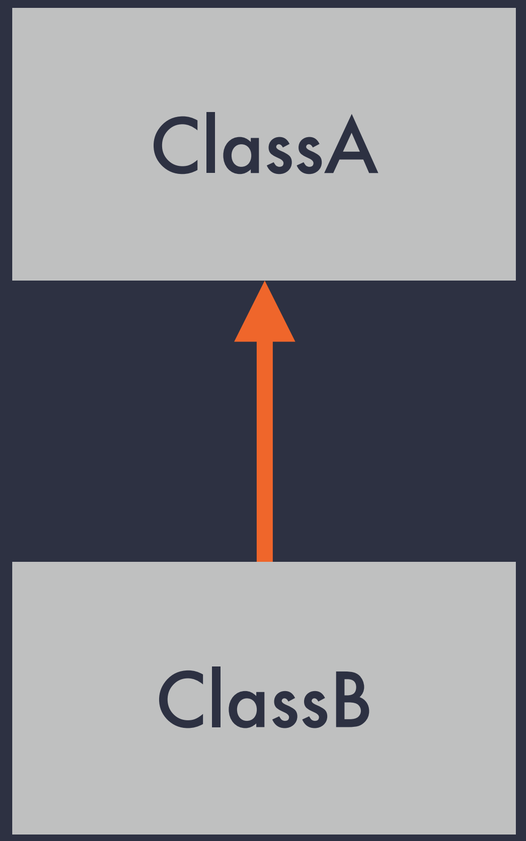

InterfaceA
Parent, base interface
inherits from
InterfaceB
Child, derived interface

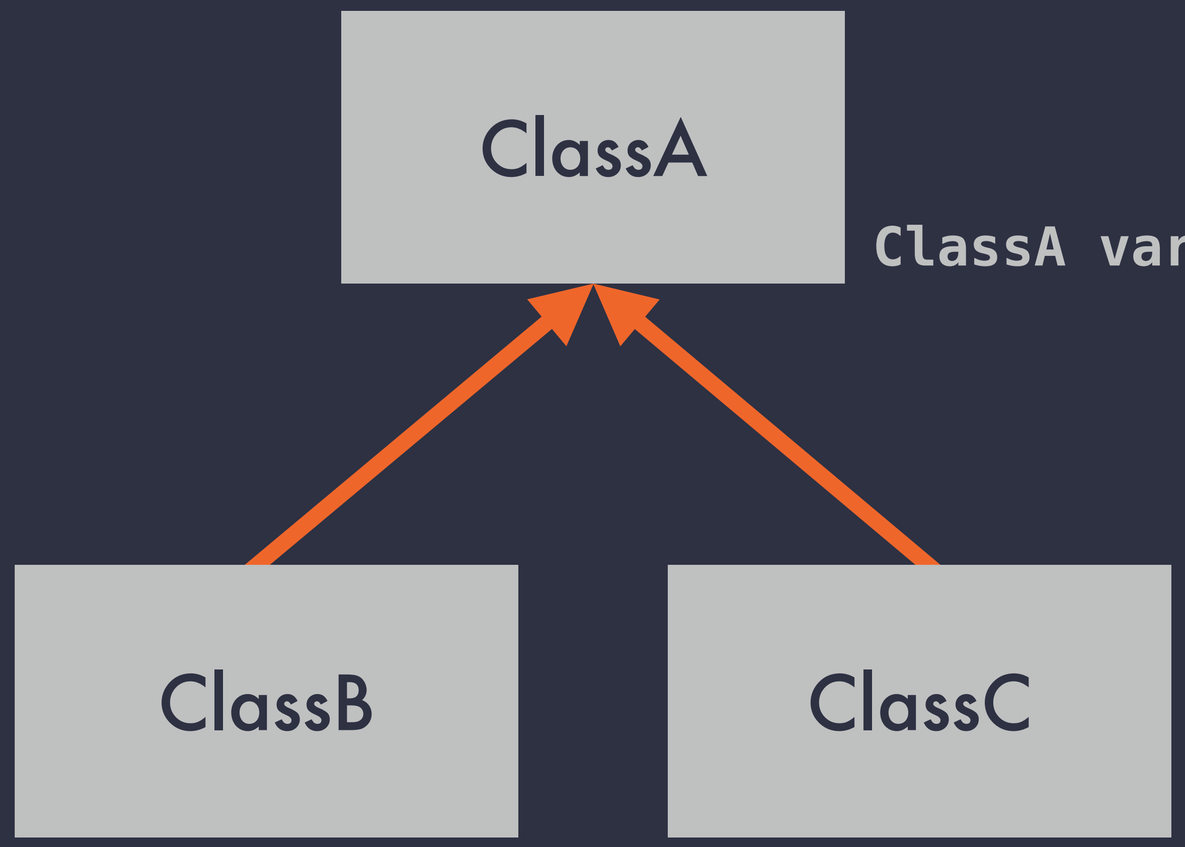

Interface
implements
Class

Item 18: Favor composition over inheritance

Effective Java, Third Edition
Joshua Bloch

Polymorphism = many forms

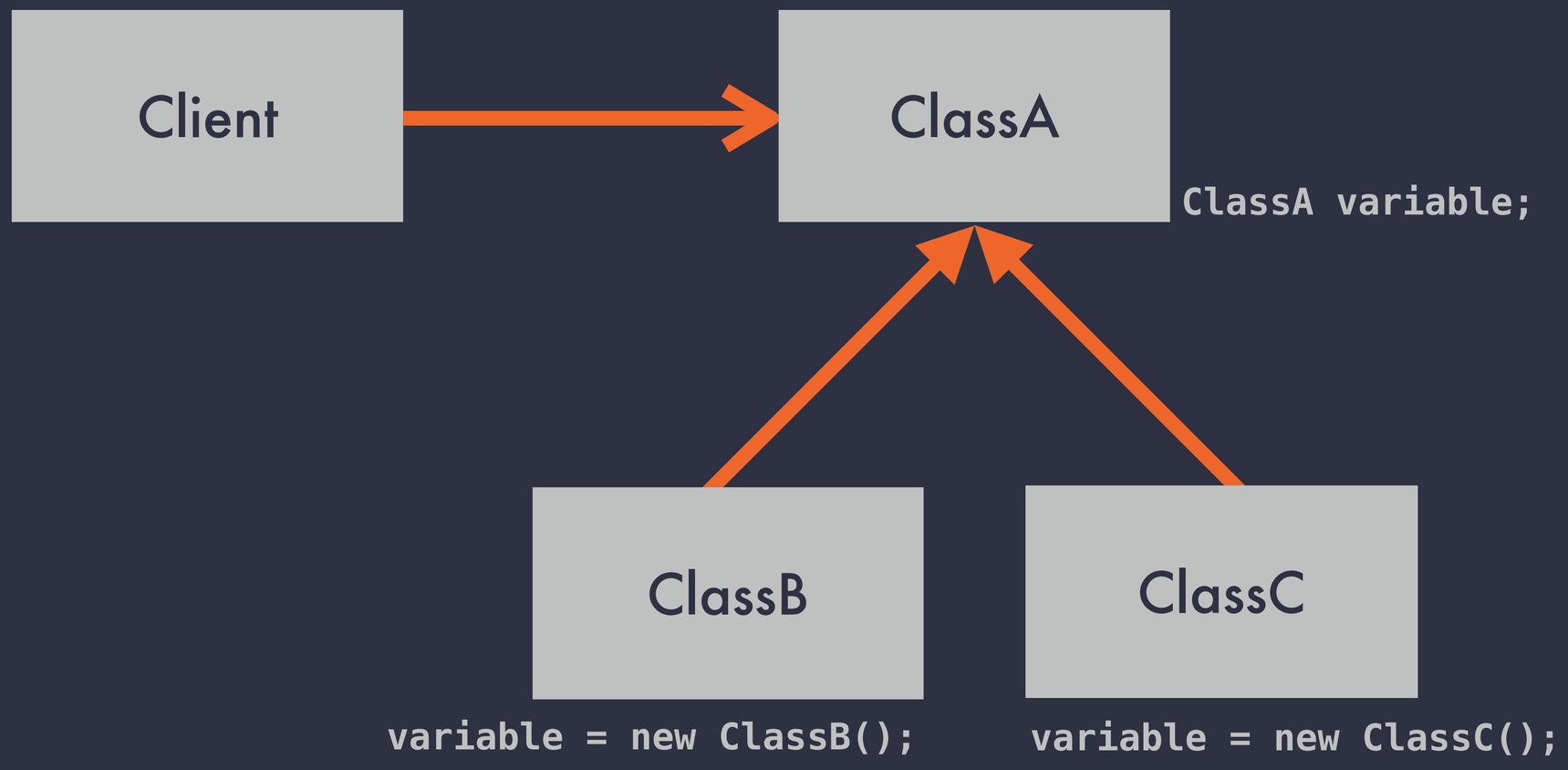

ClassA
ClassB




```
ClassA variable;
```

```
variable = new ClassB();
```

```
variable = new ClassC();
```


Client
ClassA
ClassA variable;
ClassB
ClassC
variable = new ClassB();
variable = new ClassC();

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

Meter serial number = 4512

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am

Night Plan

Period of time

[start, end)

Meter serial number

Integer >= 0

Invoice Generator

A reusable component

Invoice, with total amount

and a list of invoice lines

One invoice line for each rate

A console application

Print generated invoice with its invoice lines on the screen

* Make the design ready for another electricity consumption plan.

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

Meter serial number = 4512

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am

Night Plan

Period of time

[start, end)

Meter serial number

Integer >= 0

Invoice Generator

A reusable component

Invoice, with total amount

and a list of invoice lines

One invoice line for each rate

A console application

Print generated invoice with its invoice lines on the screen

* Make the design ready for another electricity consumption plan.

```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

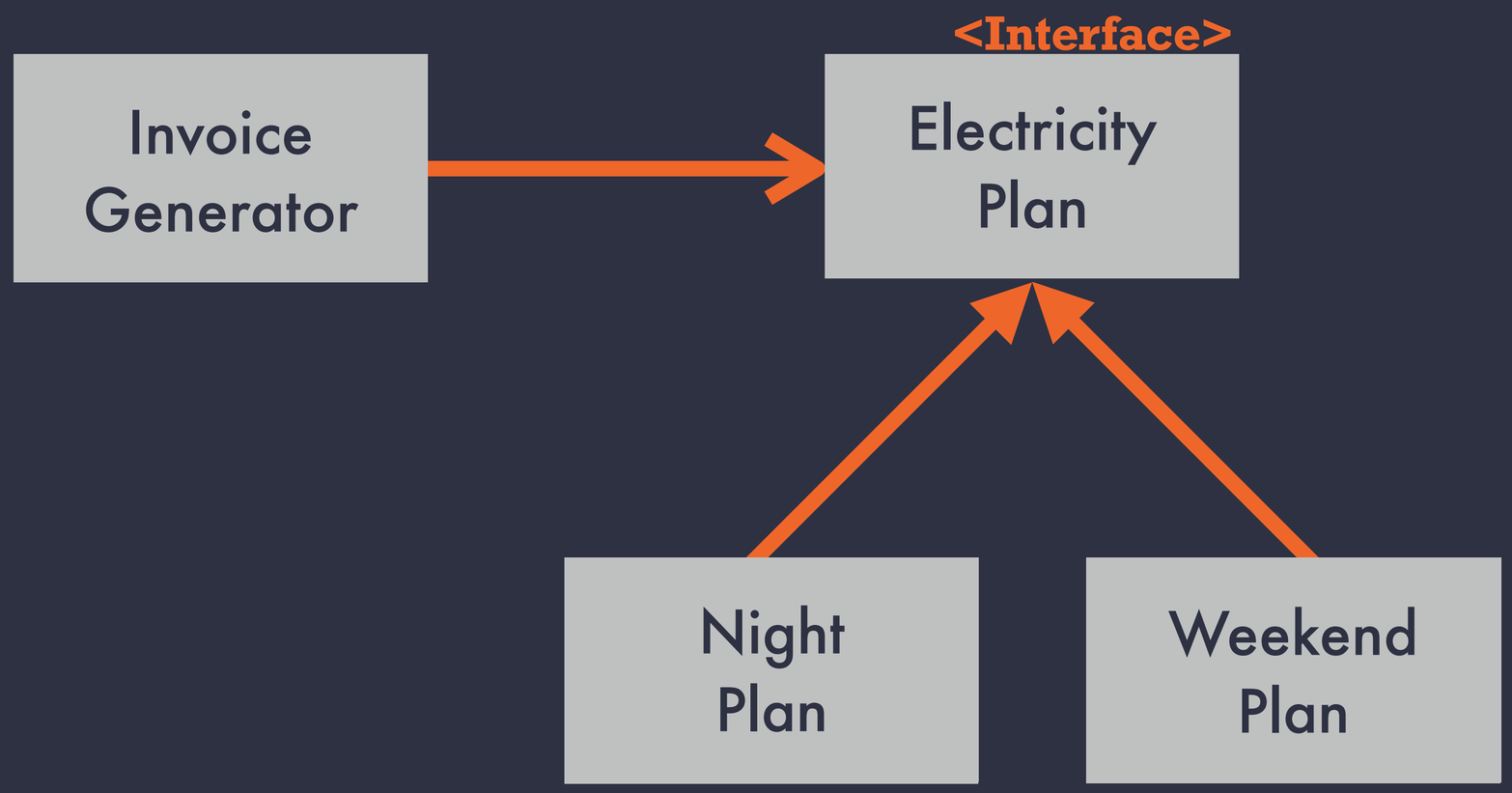
```
interface ElectricityPlan {

    void addReadingValues(List<ReadingValue> readingValues);

    List<InvoiceLine> toInvoiceLines();
}
```

```
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId,
                            ElectricityPlan electricityPlan) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

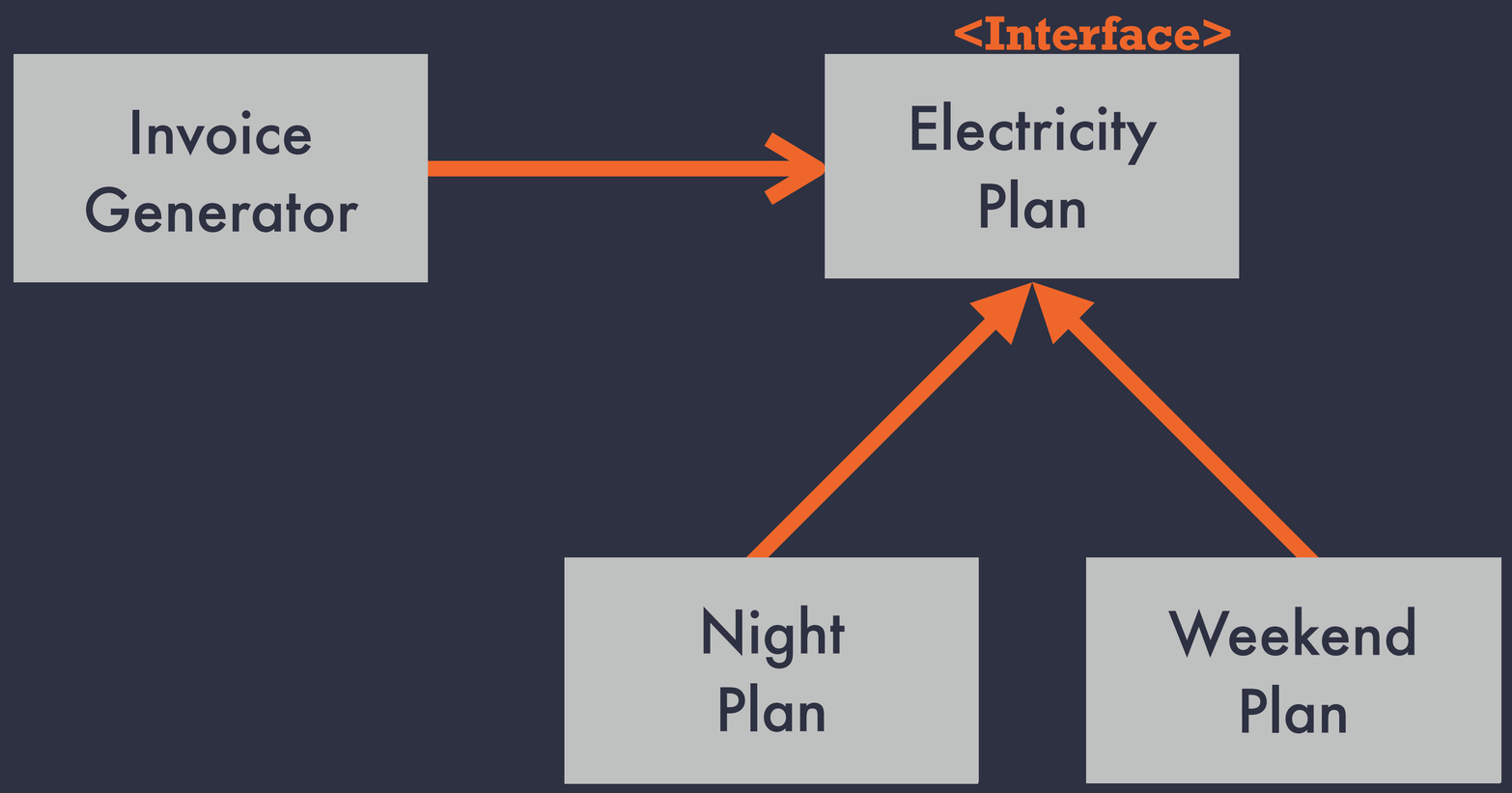
```
interface ElectricityPlan {

    void addReadingValues(List<ReadingValue> readingValues);

    List<InvoiceLine> toInvoiceLines();
}
```

```
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId,
                            ElectricityPlan electricityPlan) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        electricityPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = electricityPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

```
class NightPlan implements ElectricityPlan {
    private static final int DAY_START = 8;
    private static final int DAY_END = 23;
    private static final BigDecimal DAY_RATE = new BigDecimal("0.19288");
    private static final BigDecimal NIGHT_RATE = new BigDecimal("0.17531");

    private BigDecimal dayUsage = ZERO;
    private BigDecimal nightUsage = ZERO;

    ...
```


<Interface>
Invoice Generator
Electricity Plan
Night Plan
Weekend Plan

Design Patterns


<Interface>
Invoice Generator
Electricity Plan
Night Plan
Weekend Plan

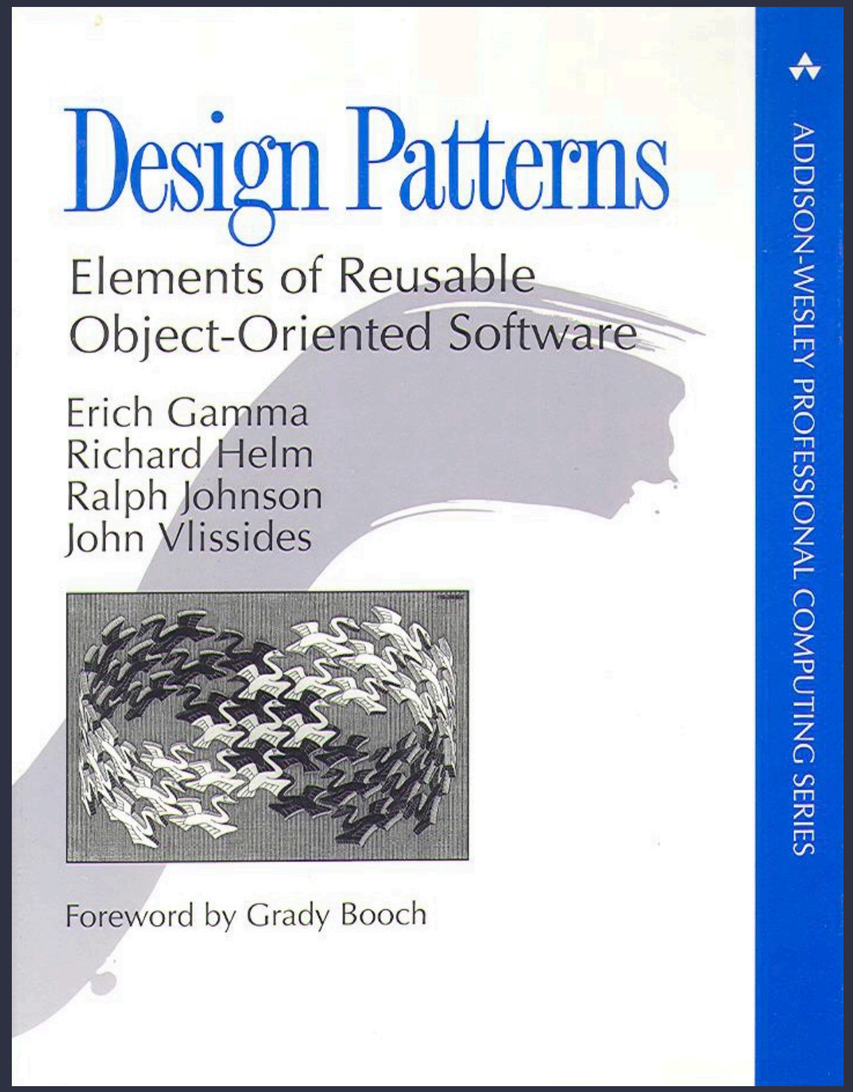
Strategy

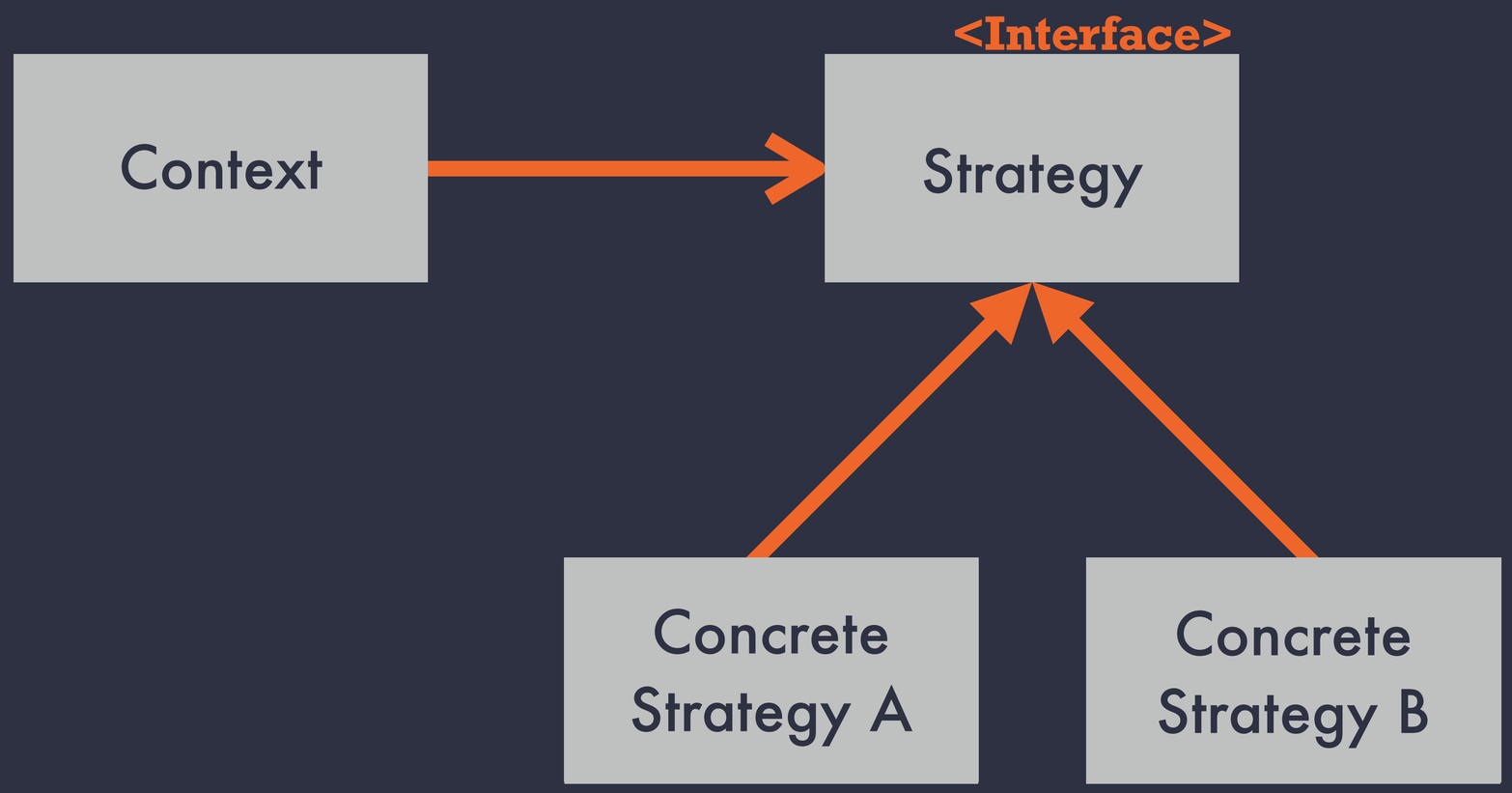
(...) a software design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design.

https://en.wikipedia.org/wiki/Software_design_pattern

Builder
Abstract Factory
Factory Method
Prototype
Singleton
Adapter
Decorator
Facade
Composite
Bridge
Proxy
Flyweight

Interpreter
Iterator
Chain of Responsibility
Mediator
Template Method
Observer
Visitor
Memento
Command
State
Strategy




<Interface>
Context
Strategy
Concrete Strategy A
Concrete Strategy B

Managing Dependencies

Cohesion

Coupling

High **Cohesion**

Low **Coupling**

```java
class NightPlan {
    private static final int DAY_START = 8;
    private static final int DAY_END = 23;
    private static final BigDecimal DAY_RATE = new BigDecimal("0.19288");
    private static final BigDecimal NIGHT_RATE = new BigDecimal("0.17531");

    private BigDecimal dayUsage = ZERO;
    private BigDecimal nightUsage = ZERO;

    void addReadingValues(List<ReadingValue> readingValues) {
        readingValues.forEach(this::addReadingValue);
    }

    private void addReadingValue(ReadingValue readingValue) {
        if (readingValue.getTimestamp().getHour() >= DAY_START
                && readingValue.getTimestamp().getHour() < DAY_END) {
            dayUsage = dayUsage.add(readingValue.getValue());
        } else {
            nightUsage = nightUsage.add(readingValue.getValue());
        }
    }

    List<InvoiceLine> toInvoiceLines() {
        ArrayList<InvoiceLine> invoiceLines = new ArrayList<>();
        InvoiceLine dayInvoiceLine = new InvoiceLine("Day Rate", DAY_RATE,
                dayUsage, dayUsage.multiply(DAY_RATE));
        InvoiceLine nightInvoiceLine = new InvoiceLine("Night Rate", NIGHT_RATE,
                nightUsage, nightUsage.multiply(NIGHT_RATE));
        invoiceLines.add(dayInvoiceLine);
        invoiceLines.add(nightInvoiceLine);
        return invoiceLines;
    }
}
```

```
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

High **Cohesion**

Low **Coupling**

Single Responsibility Principle

Open-Closed Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

Single Responsibility Principle

One reason to change

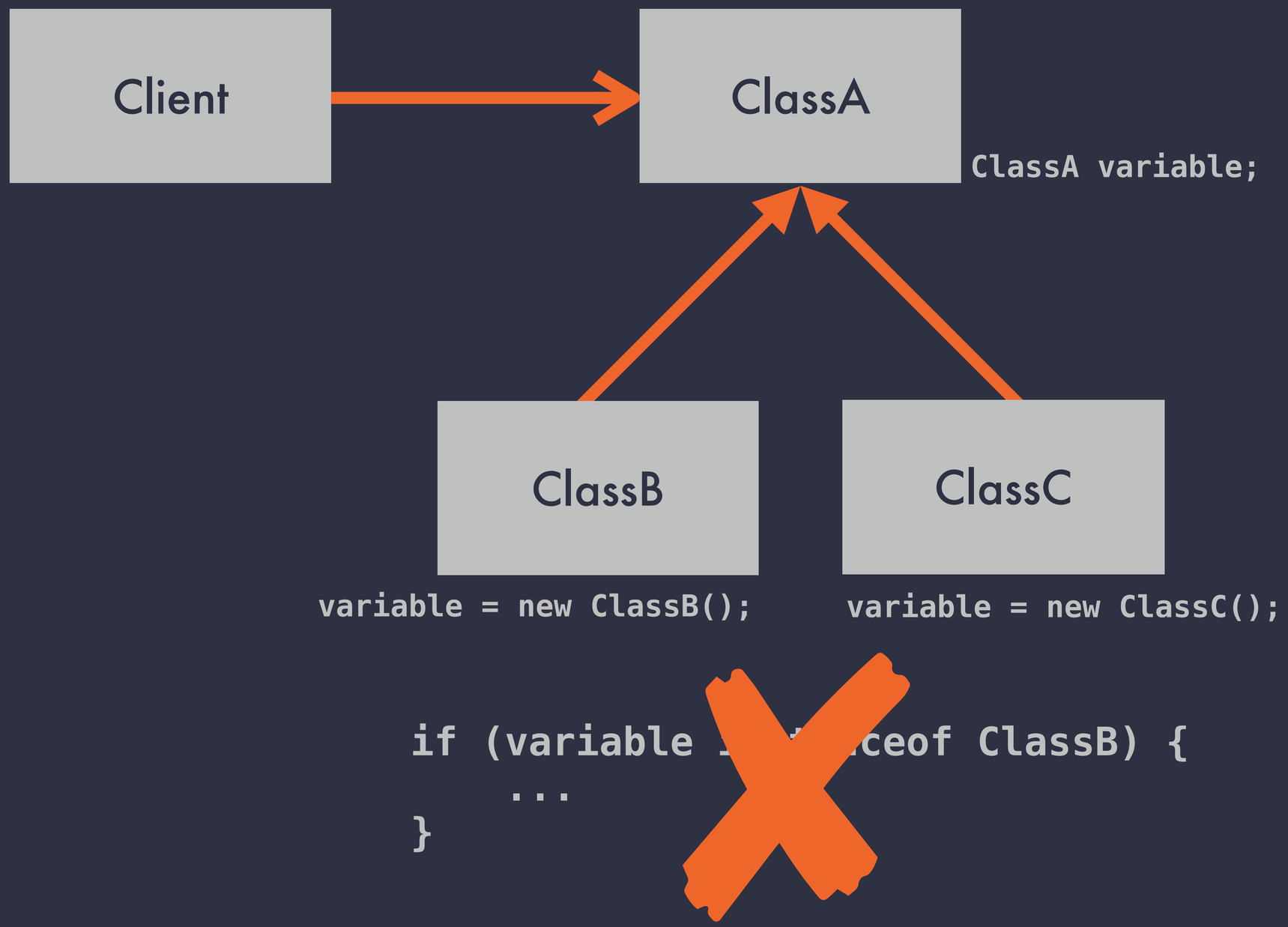
```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

Open-Closed Principle

Open for extension, closed for modification

```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId,
                            ElectricityPlan electricityPlan) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        electricityPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = electricityPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

Liskov Substitution Principle


Client
ClassA
ClassA variable;
ClassB
ClassC
variable = new ClassB();
variable = new ClassC();
if (variable
ceof ClassB) {
...
}

Interface Segregation

Dependency Inversion

High level concepts should not depend on low level concepts.

Both should depend on abstraction.

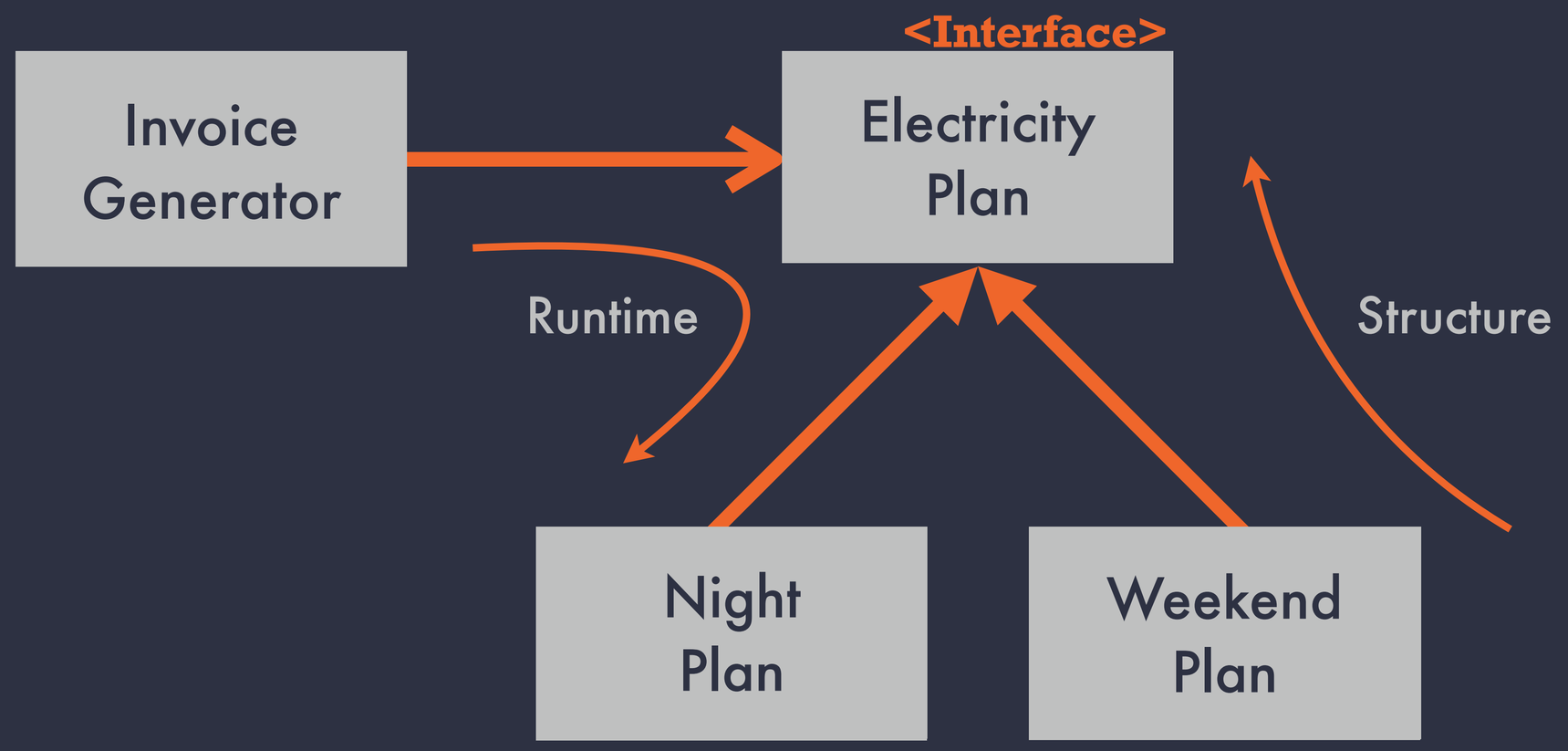
```java
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        NightPlan nightPlan = new NightPlan();
        nightPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = nightPlan.toInvoiceLines();
        return new Invoice(invoiceLines);
    }
}
```

```
class InvoiceGenerator {
    private final ReadingValuesGenerator readingValuesGenerator;

    InvoiceGenerator(ReadingValuesGenerator readingValuesGenerator) {
        this.readingValuesGenerator = readingValuesGenerator;
    }

    Invoice generateInvoice(LocalDateTime sinceClosed, LocalDateTime untilOpen, int meterId,
                            ElectricityPlan electricityPlan) {
        List<ReadingValue> readingValues = readingValuesGenerator.generate(sinceClosed, untilOpen, meterId);
        electricityPlan.addReadingValues(readingValues);
        List<InvoiceLine> invoiceLines = electricityPlan.toInvoiceLines();
        ice(invoiceLines);
    }
}
```

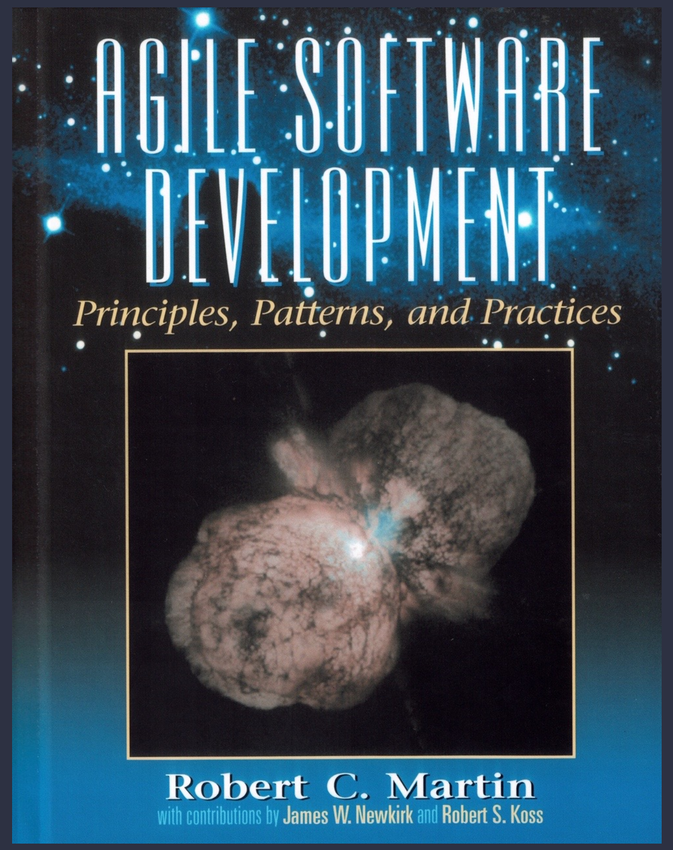
NightPlan

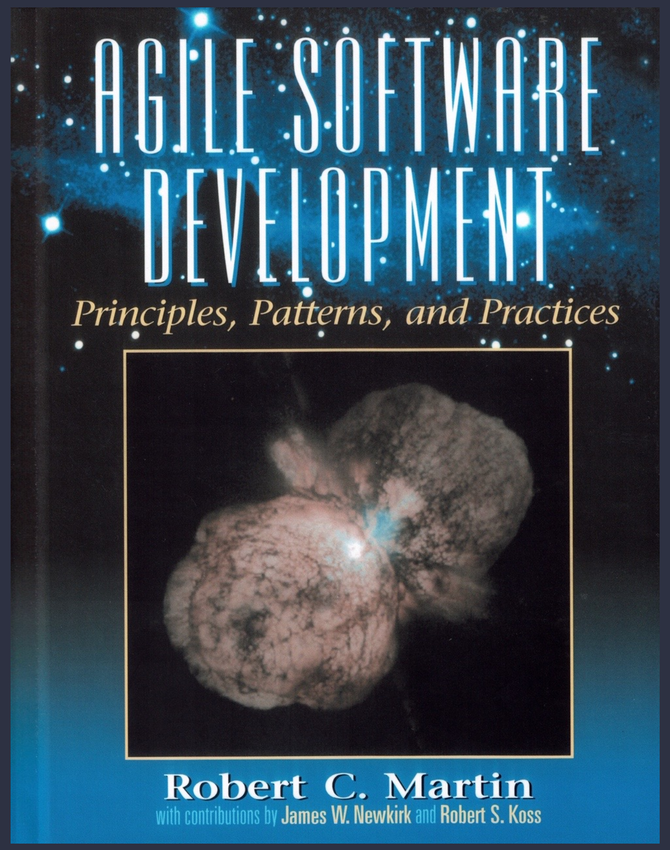
NightPlan

NightPlan



Dependency *Inversion*


AGILE SOFTWARE DEVELOPMENT
Principles, Patterns, and Practices
Robert C. Martin
with contributions by James W. Newkirk and Robert S. Koss



https://cleancoders.com

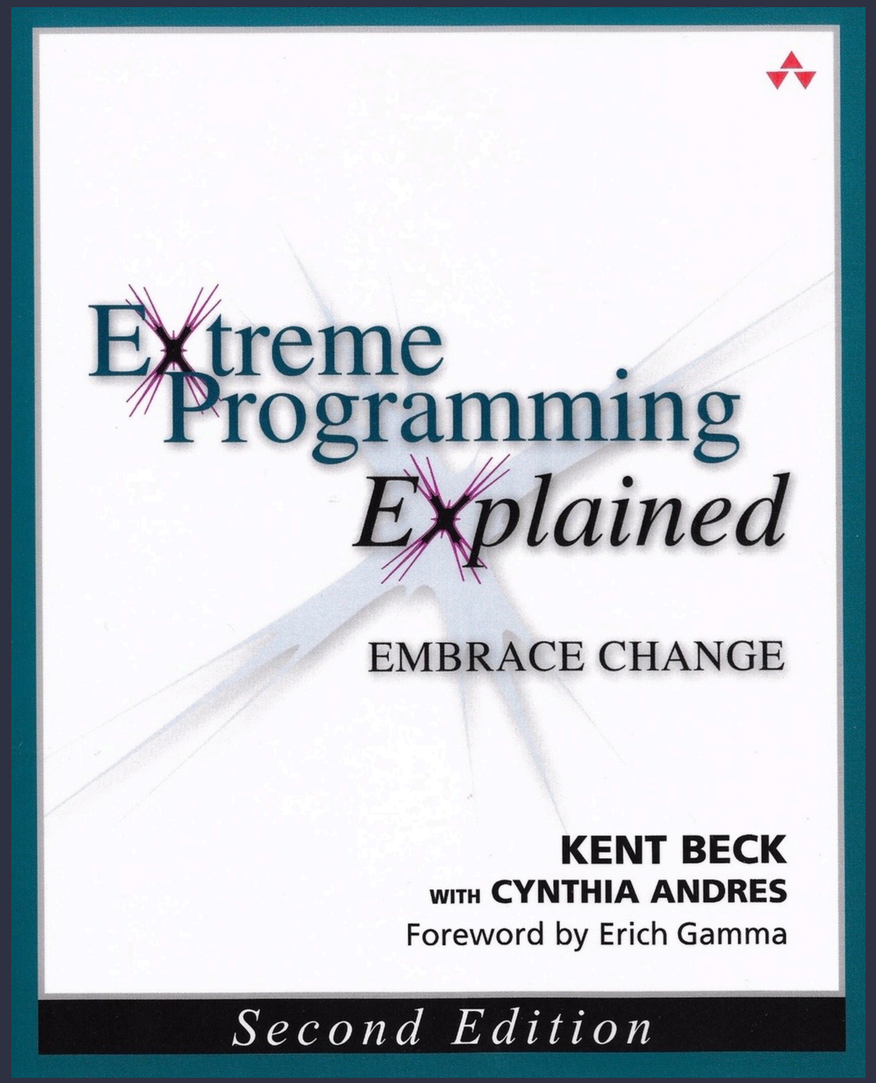
Summary

Disciplines

Scrum

Scrum

XP (Extreme Programming)


Extreme Programming Explained
EMBRACE CHANGE
KENT BECK
WITH CYNTHIA ANDRES
Foreword by Erich Gamma
Second Edition

Pair programming

Simple design

Refactoring

Testing

Simple design

≠ trivial

≠ easy

YAGNI

You Ain’t Gonna Need It

Refactoring

... as much as possible

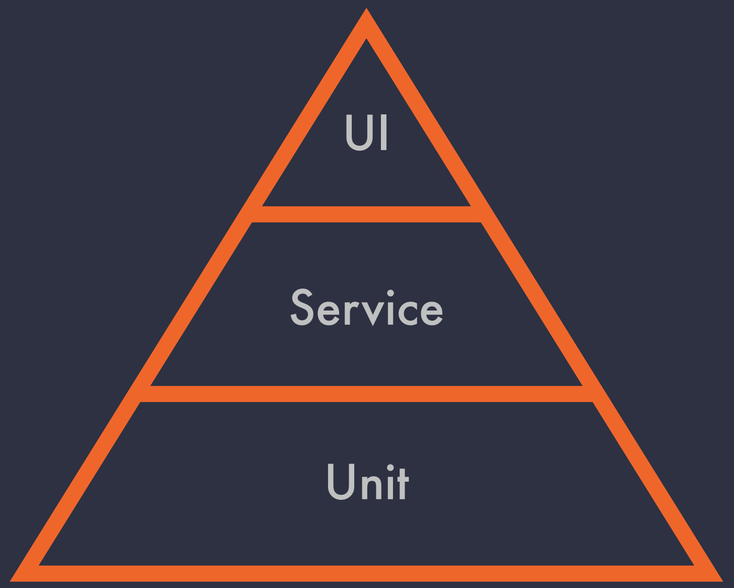
... to simplify design

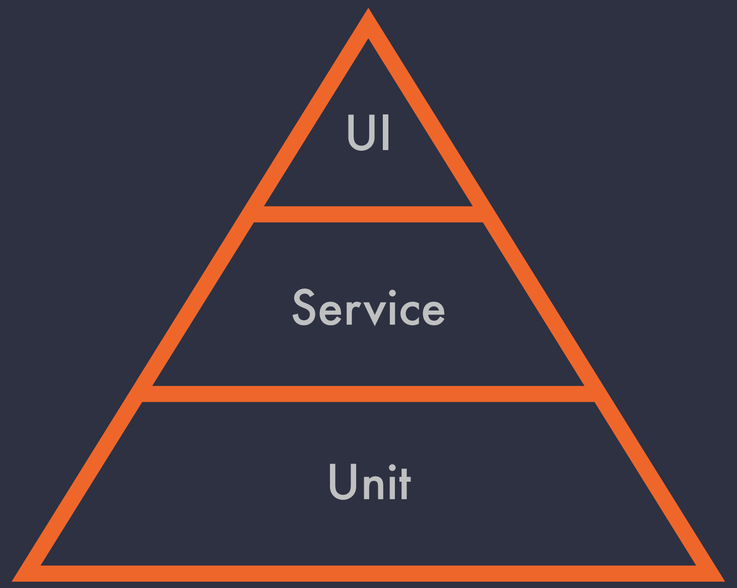
Testing

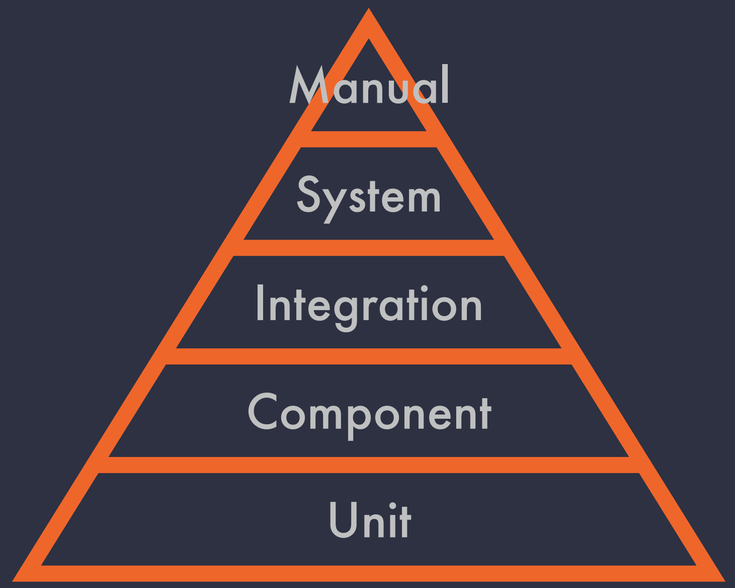
Test Pyramid

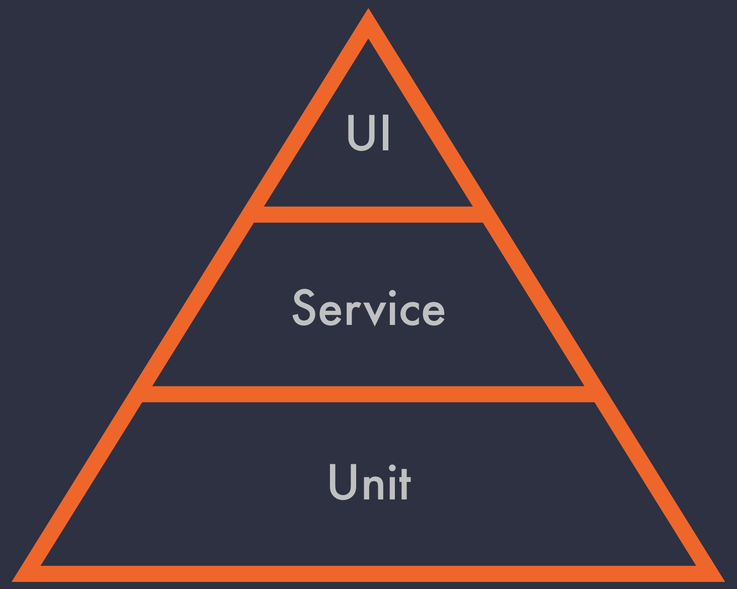
TDD

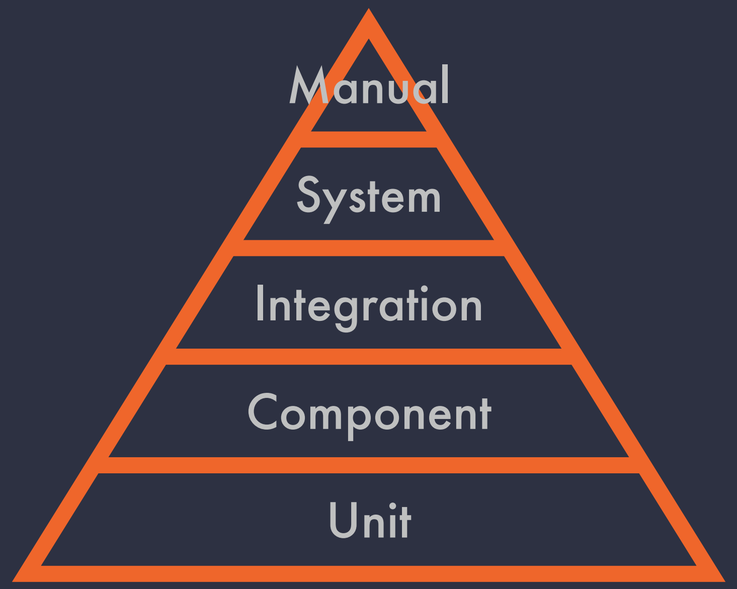
Test-Driven Development


UI
Service
Unit


UI
Service
Unit


Manual
System
Integration
Component
Unit



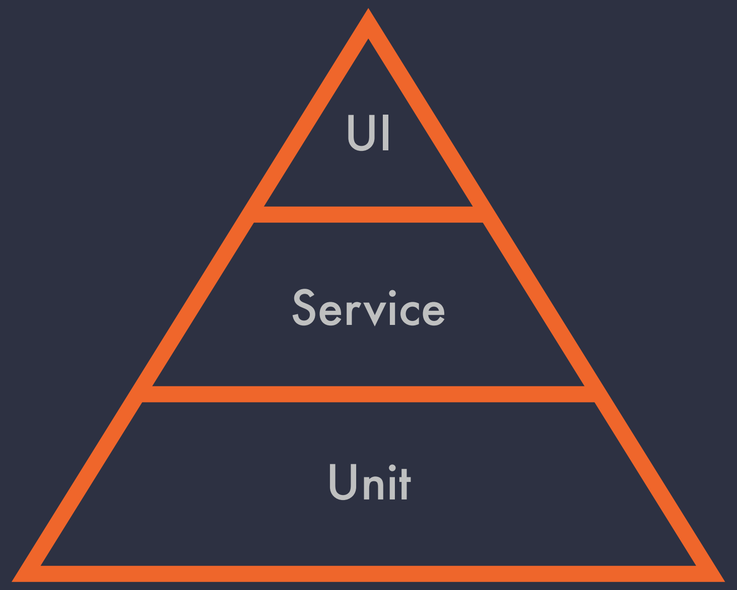


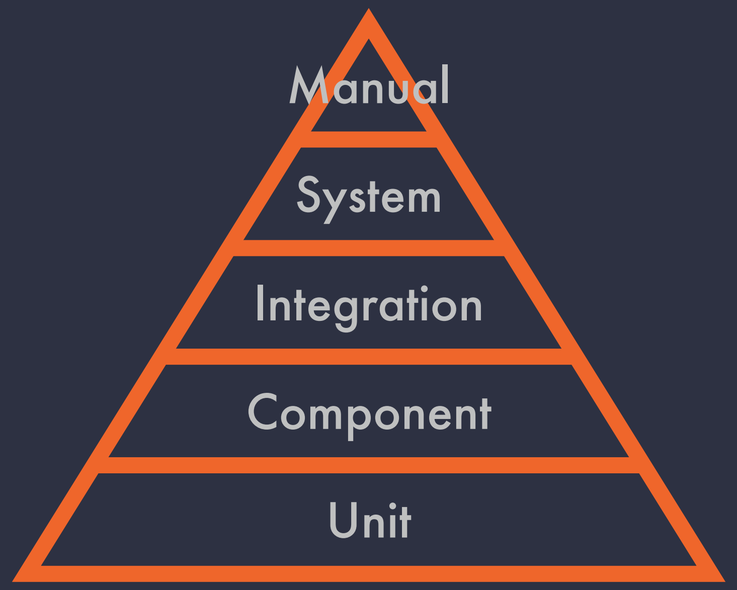
Number of tests?

Speed of execution?

Isolation vs integration?

smaller number of tests



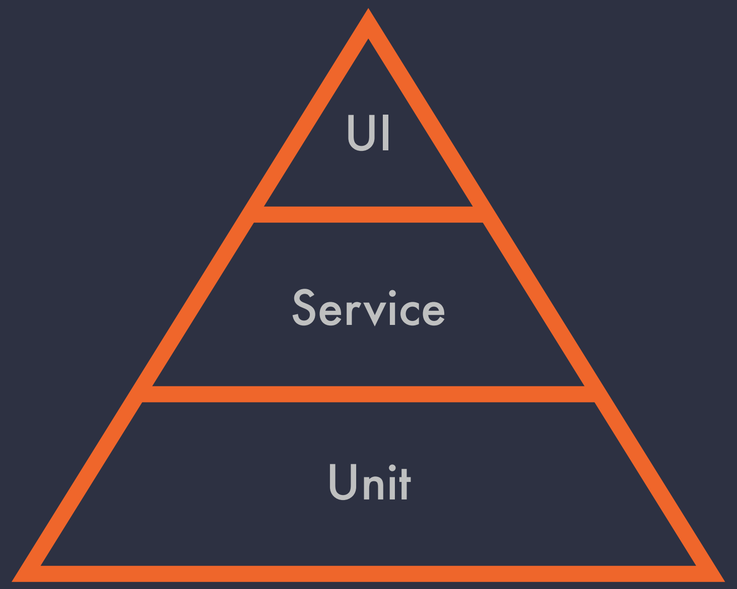


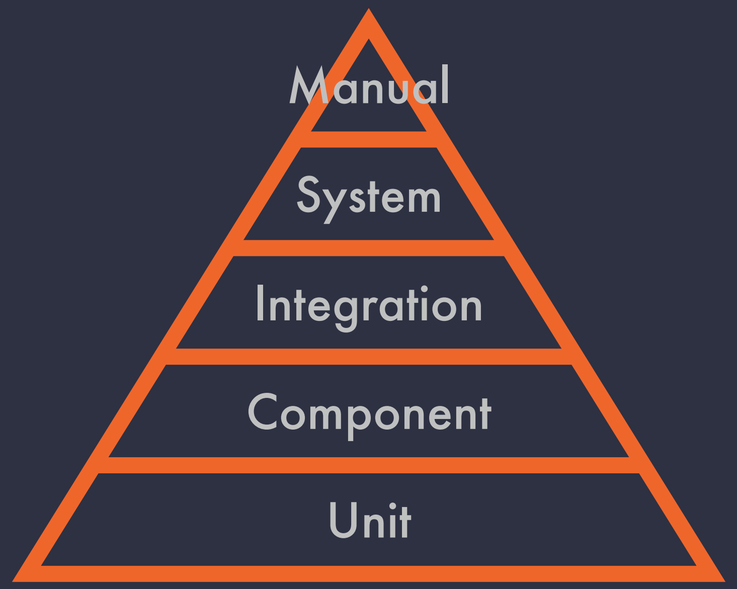
greater number of tests

Speed of execution?

Isolation vs integration?

testing integration
smaller number of tests

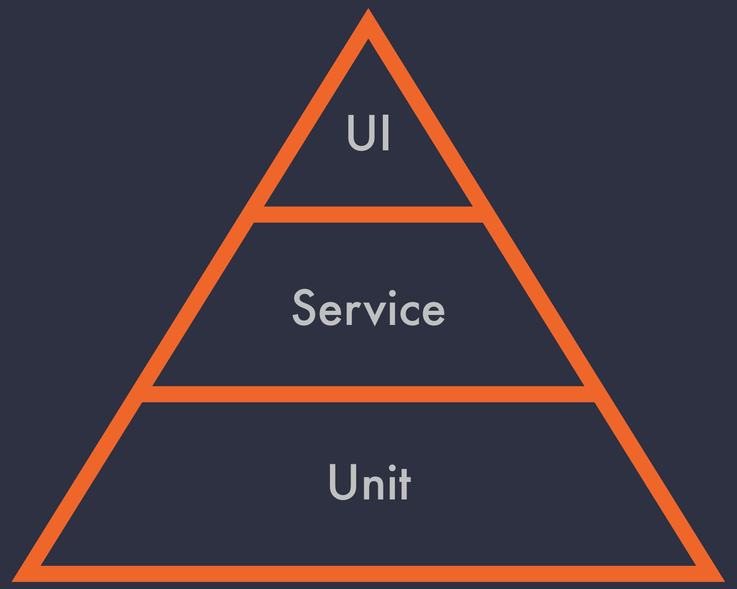


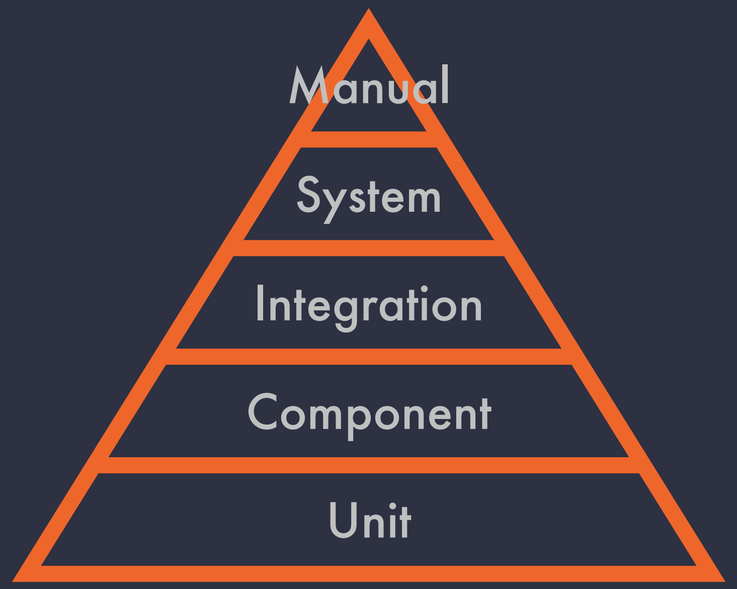


greater number of tests
testing in isolation

Speed of execution?

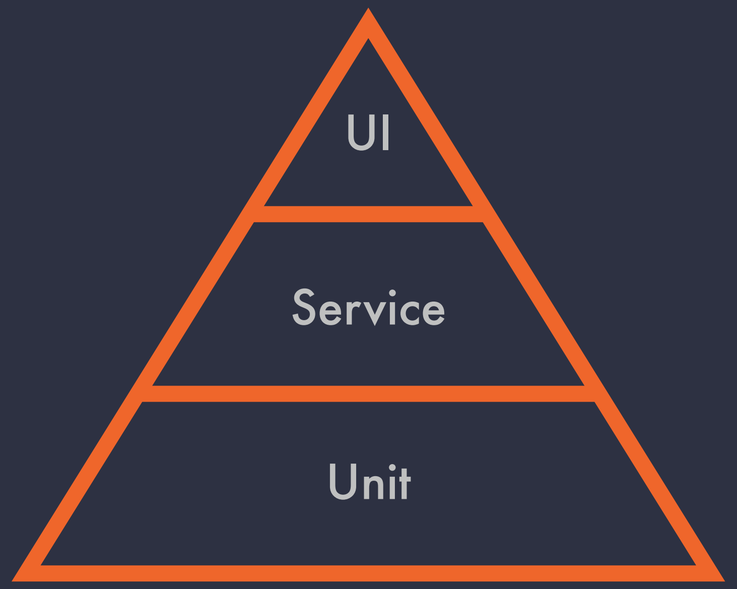
slower tests
testing integration
smaller number of tests

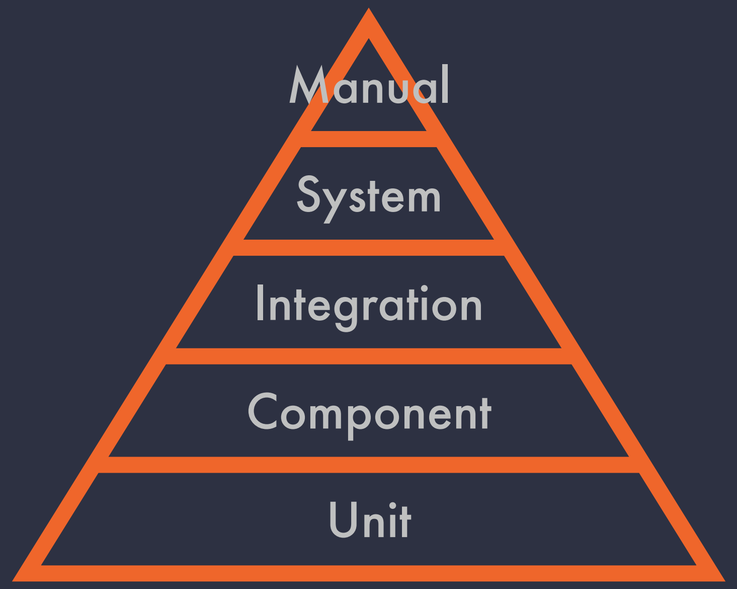




greater number of tests
testing in isolation
faster tests

slower tests
testing integration
smaller number of tests





greater number of tests
testing in isolation
faster tests

NrgyInvoicR

Unit tests

backend project

jUnit

System tests

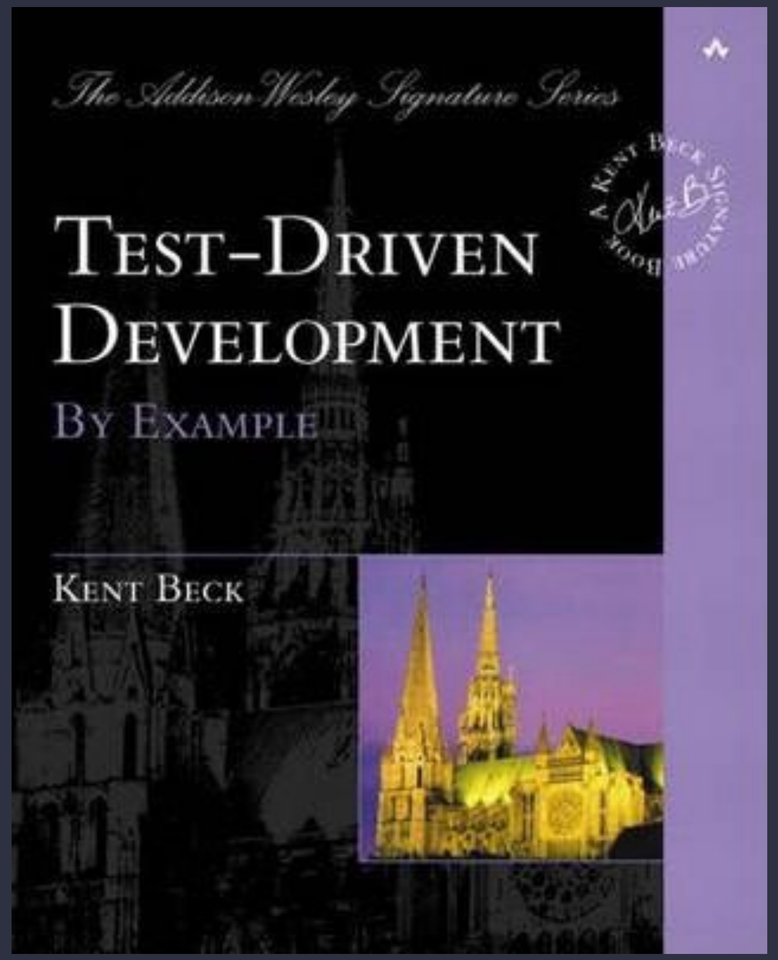
systemtesting project

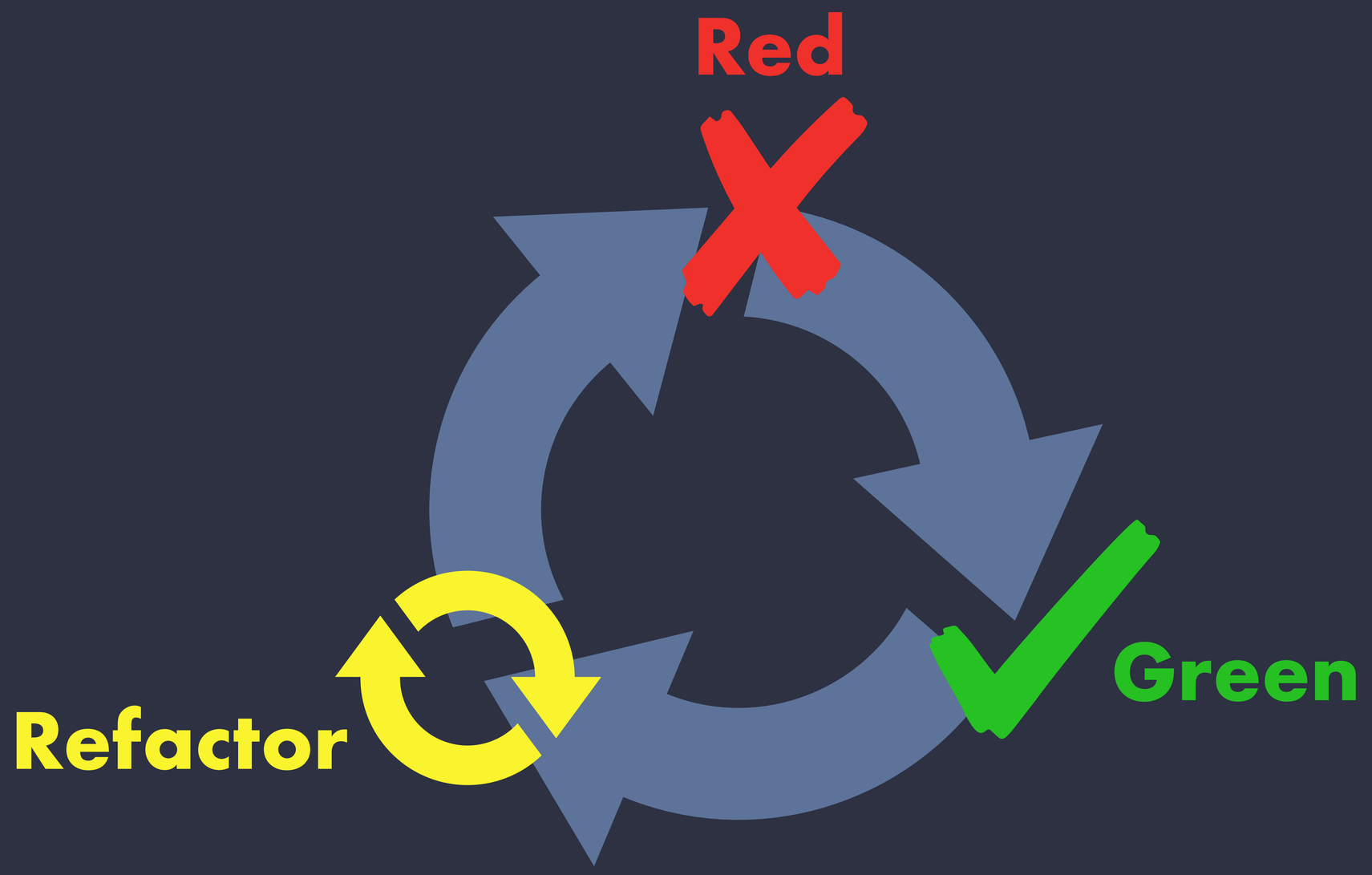
jUnit

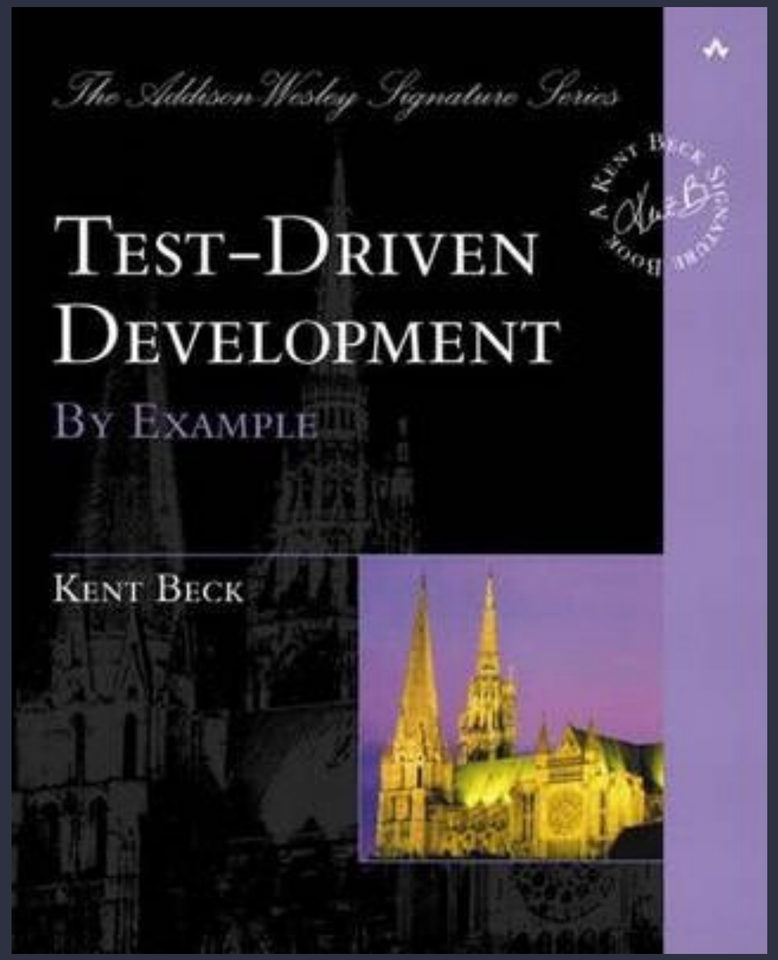
Selenium

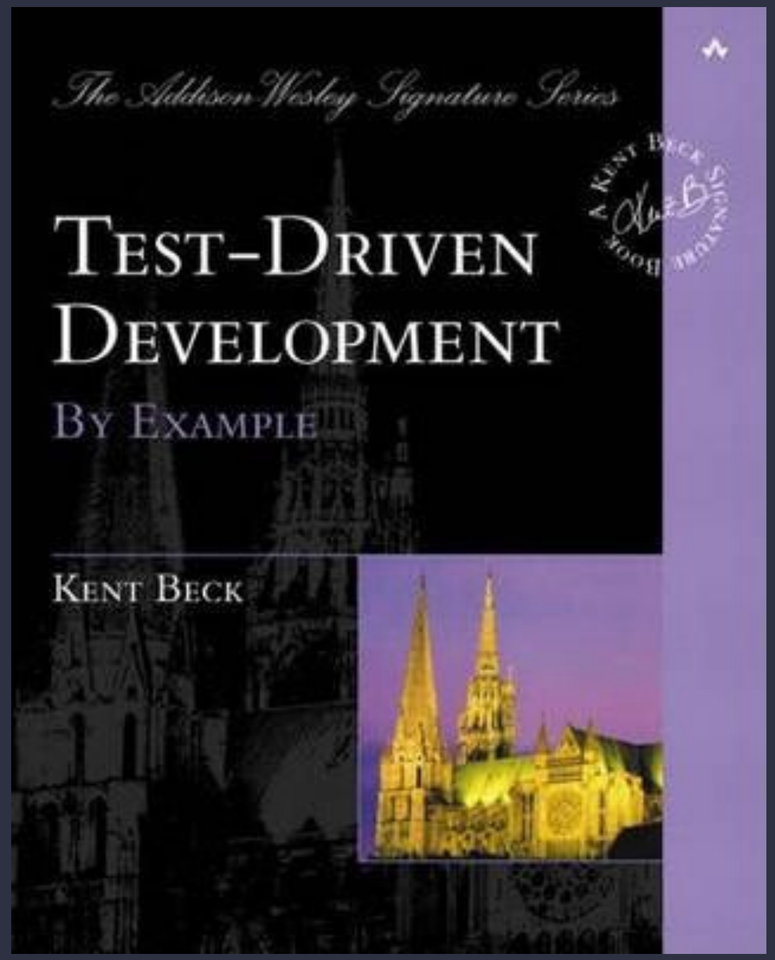
TDD

Test-Driven Development


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck


Red
Green
Refactor


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck



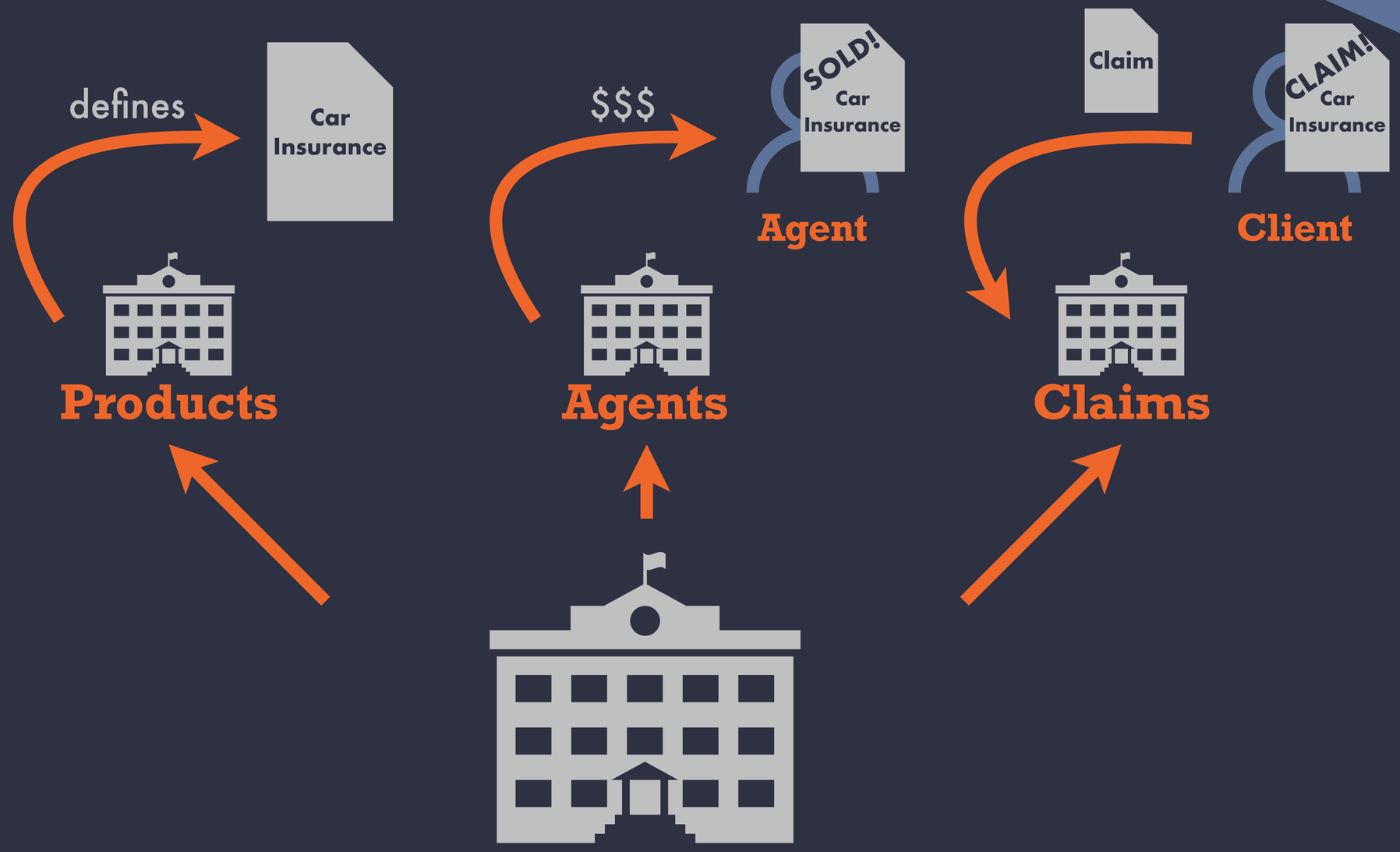
https://cleancoders.com

Summary

Complex Domains

POST	CREATE
GET	READ
PUT	UPDATE
DELETE	DELETE

Insurance Company

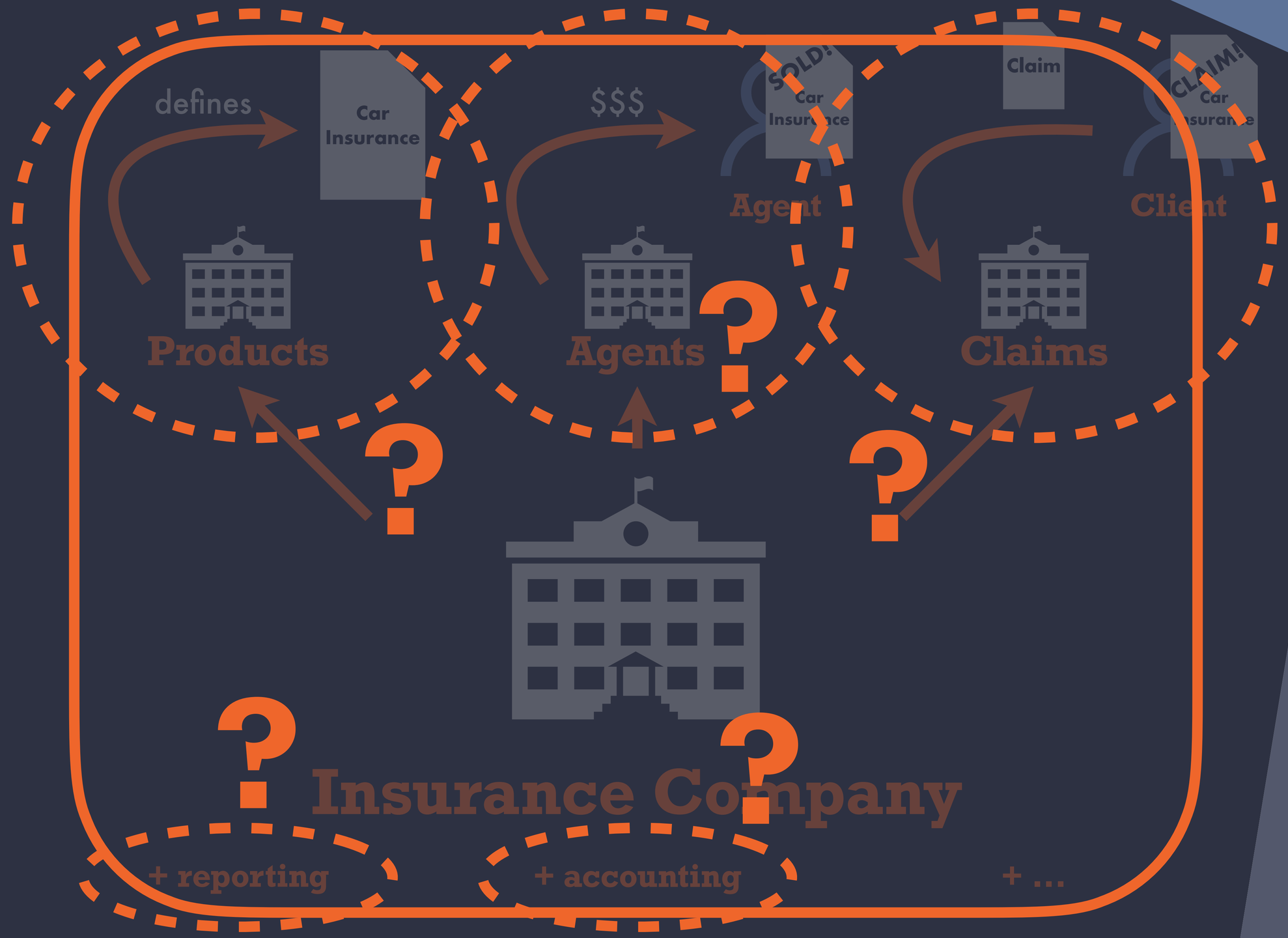


Insurance Company

+ reporting

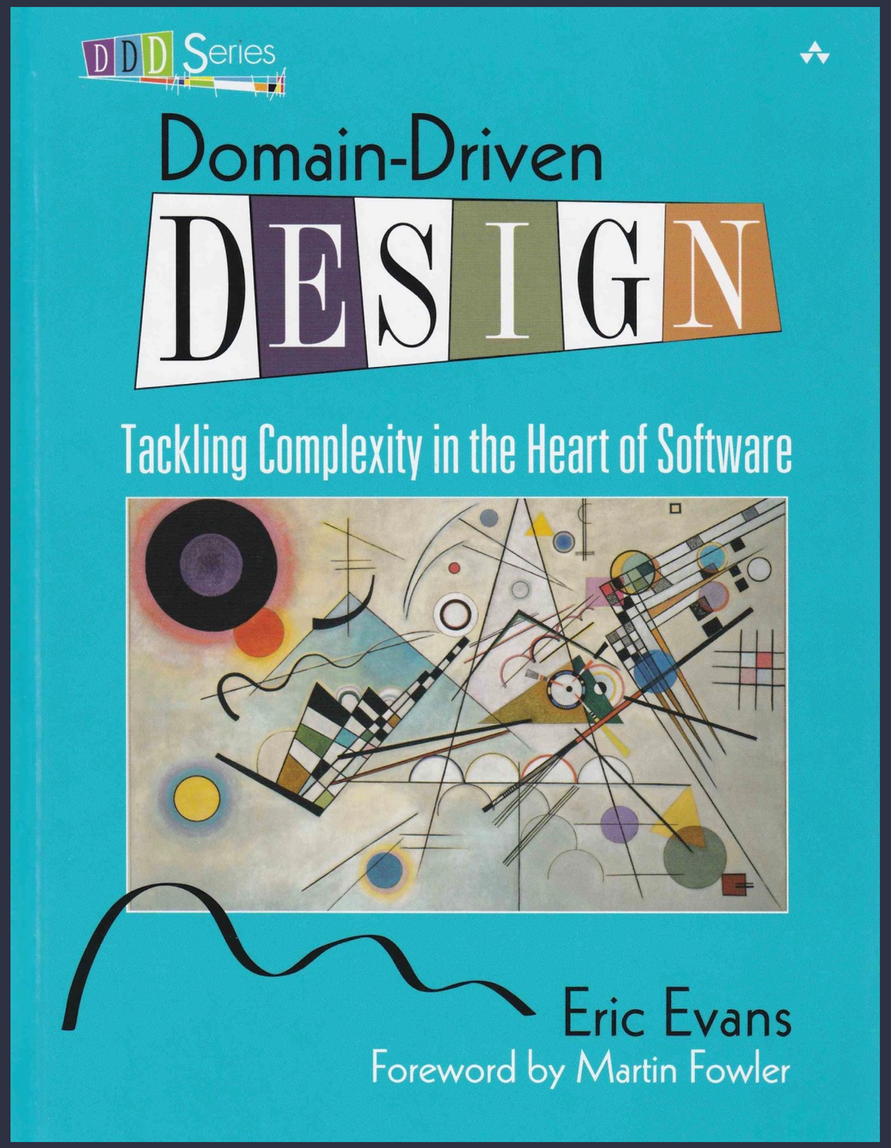
+ accounting

+ ...

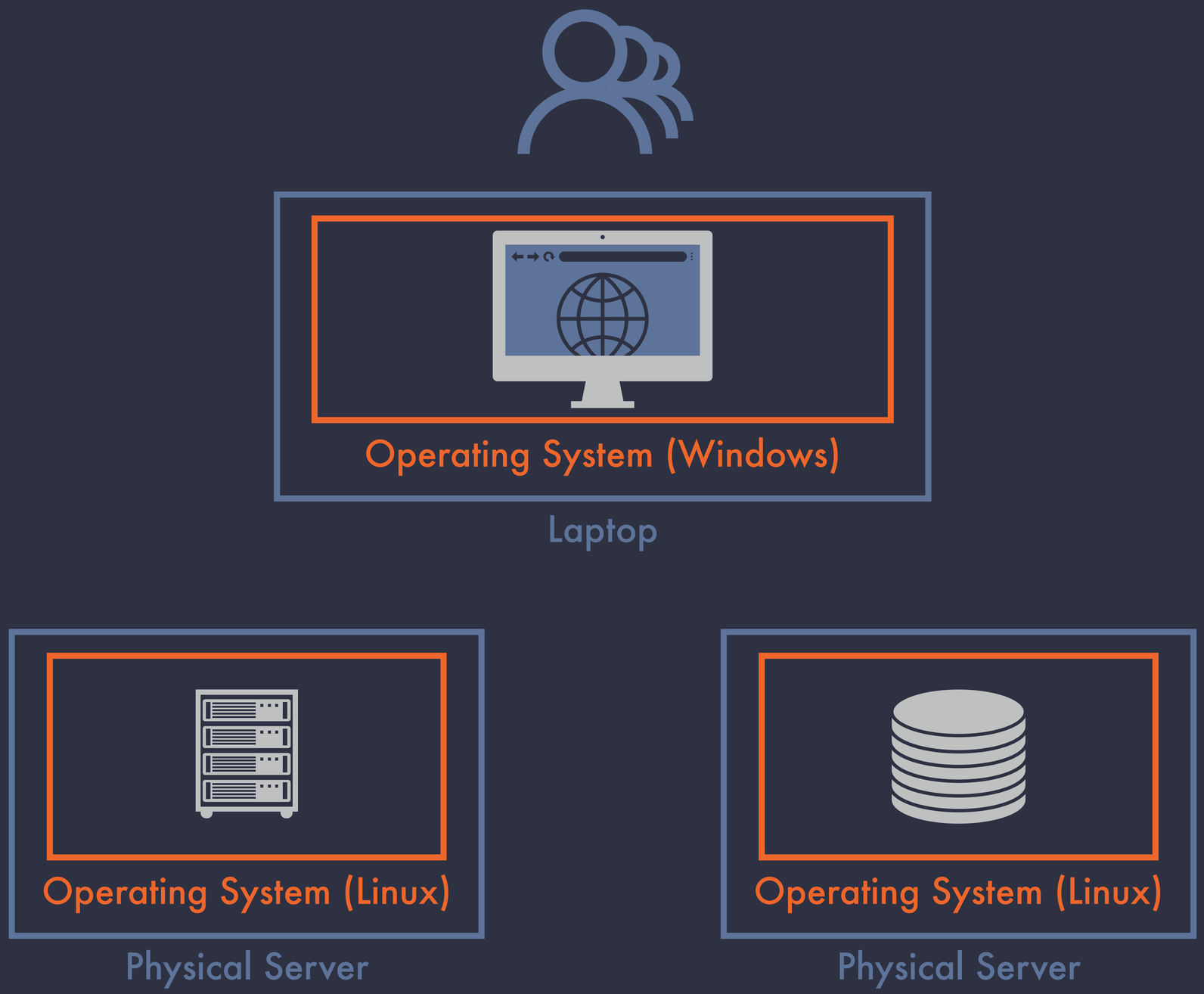

defines
Car Insurance
Products
$$$
SOLD!
Car Insurance
Agent
Agents
Claim
CLAIM!
Car Insurance
Client
Claims
Insurance Company
+ reporting
+ accounting
+ ...

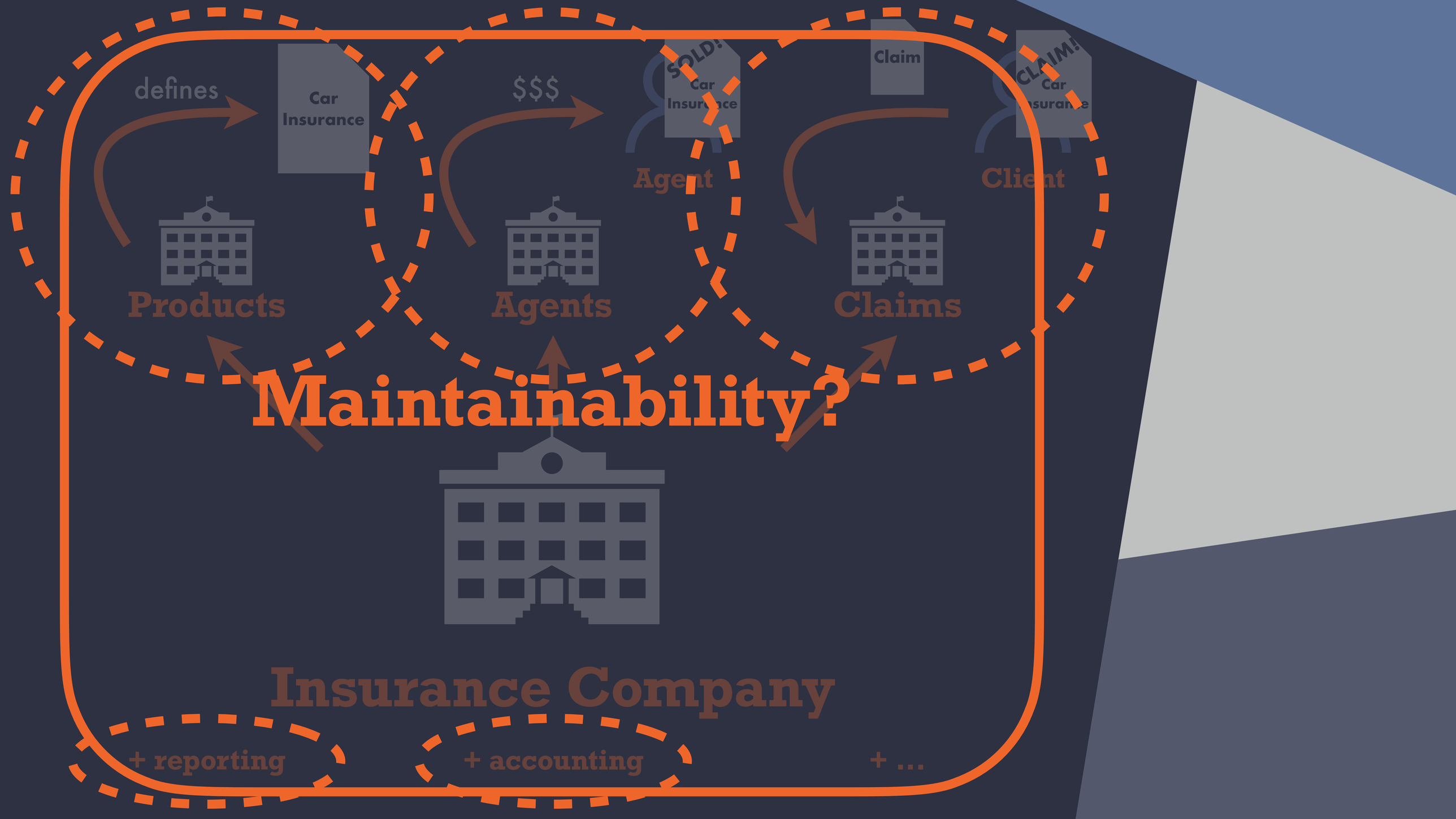
DDD

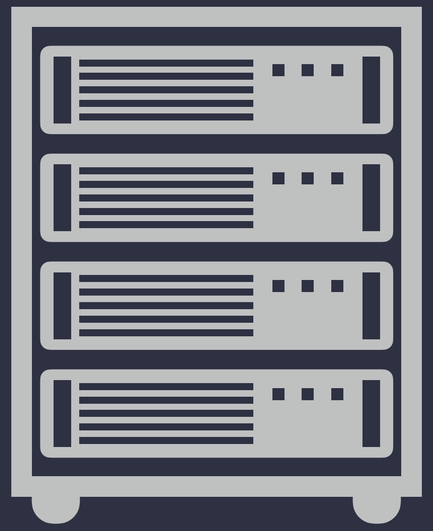
Domain-Driven Design



Systems Architecture


Operating System (Windows)
Laptop
Operating System (Linux)
Physical Server
Operating System (Linux)
Physical Server


defines
Car Insurance
$$$
SOLD!
Car Insurance
Claim
CLAIM!
Car Insurance
Agent
Client
Products
Agents
Claims
Maintainability?
Insurance Company
+ reporting
+ accounting
+ ...



Spring Application

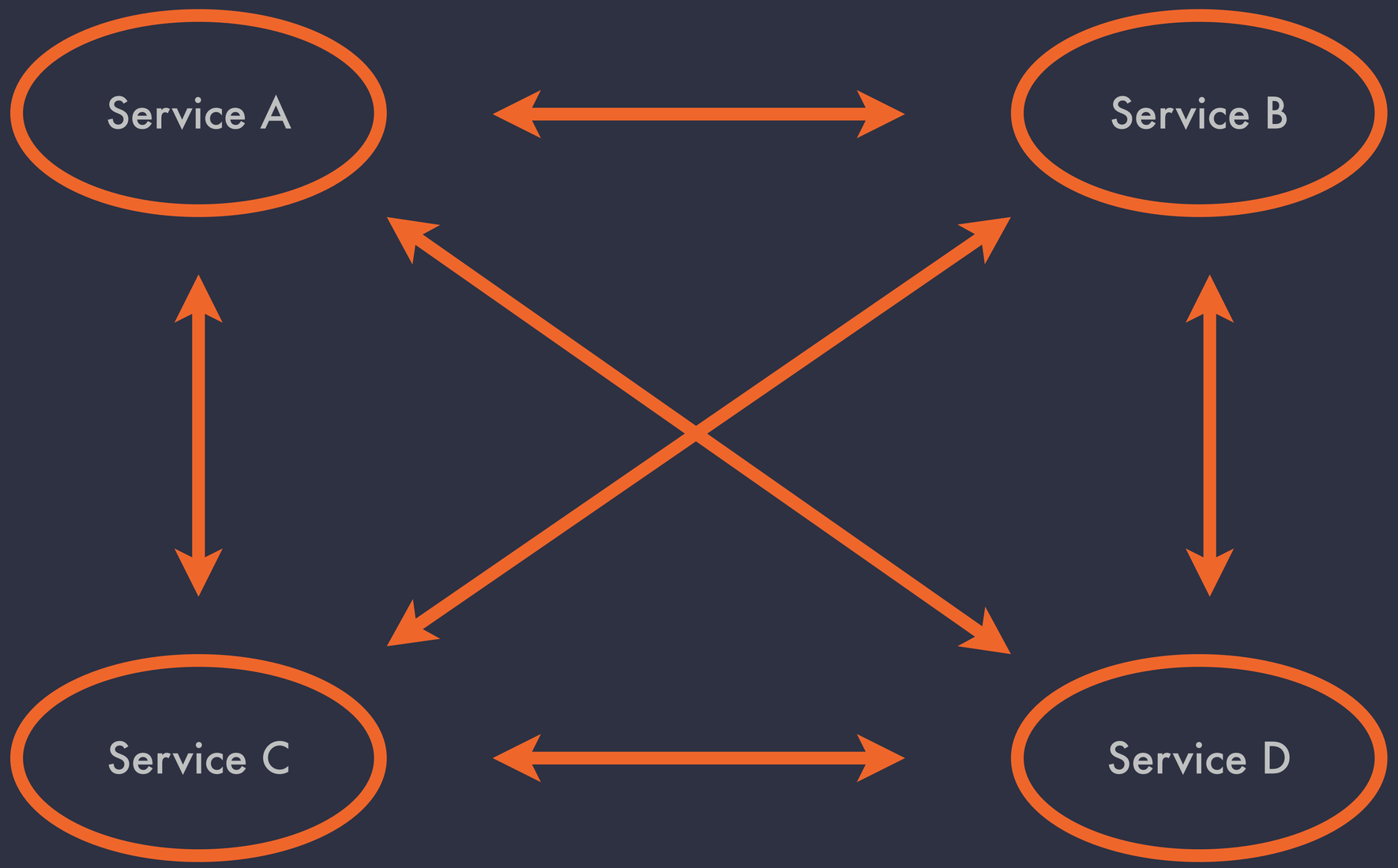
Monolith

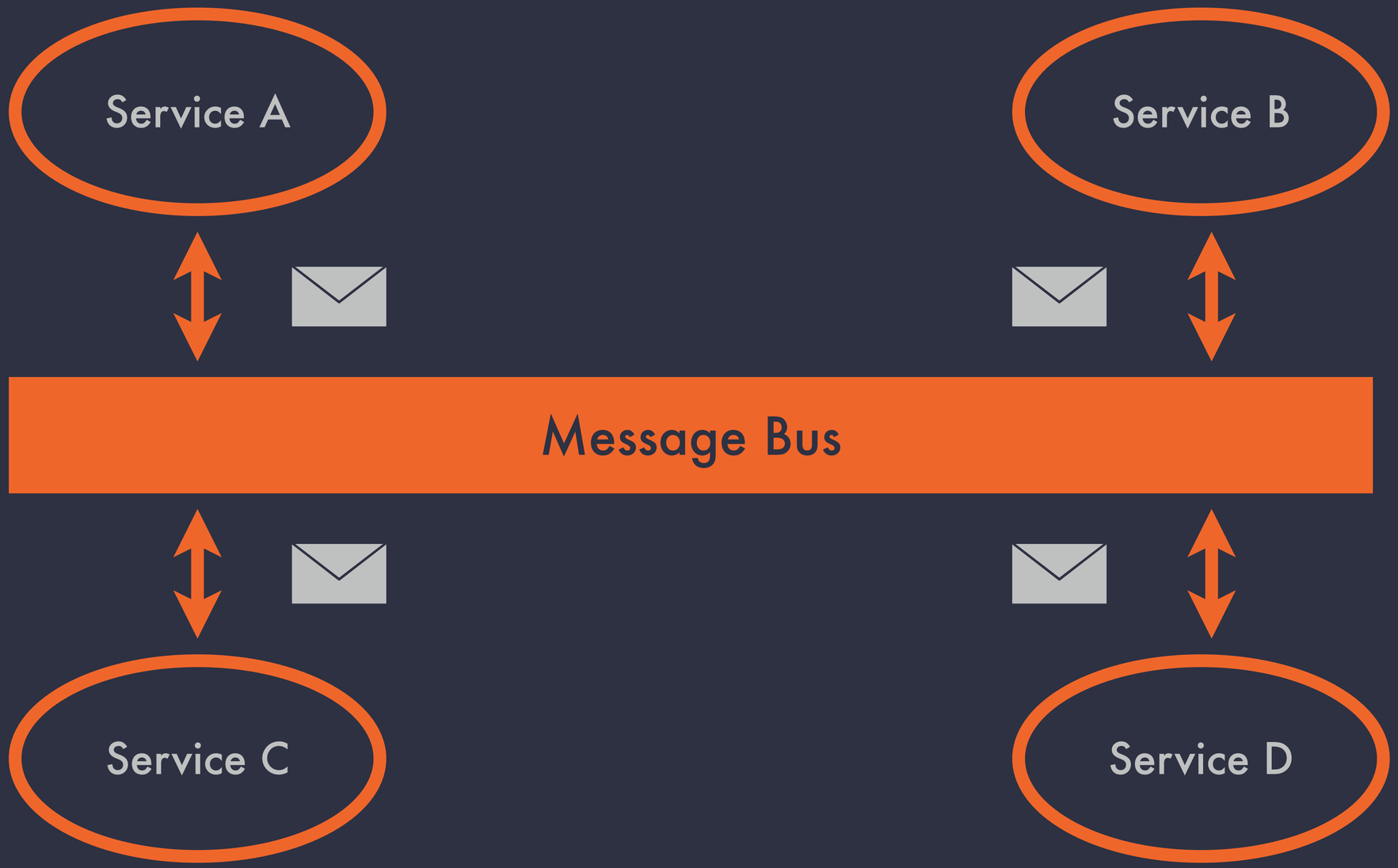
Monolithic architechture

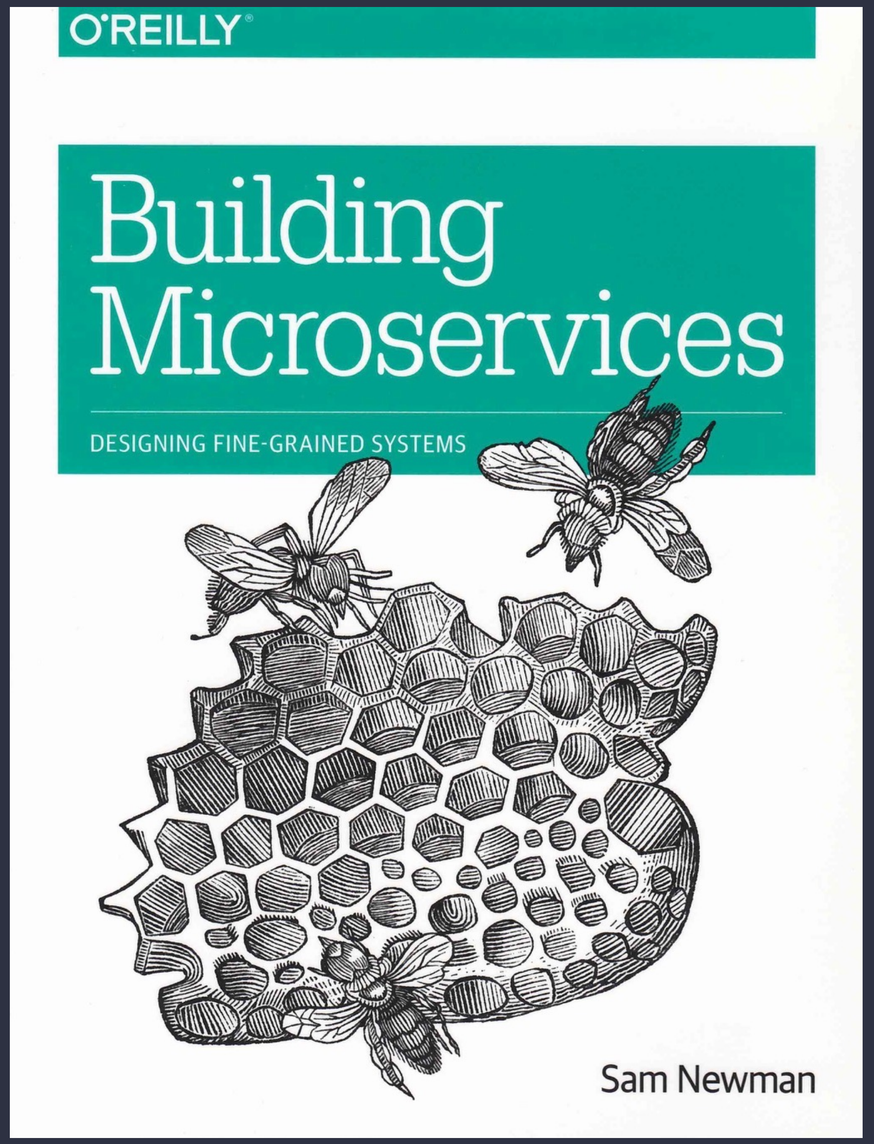
Microservices

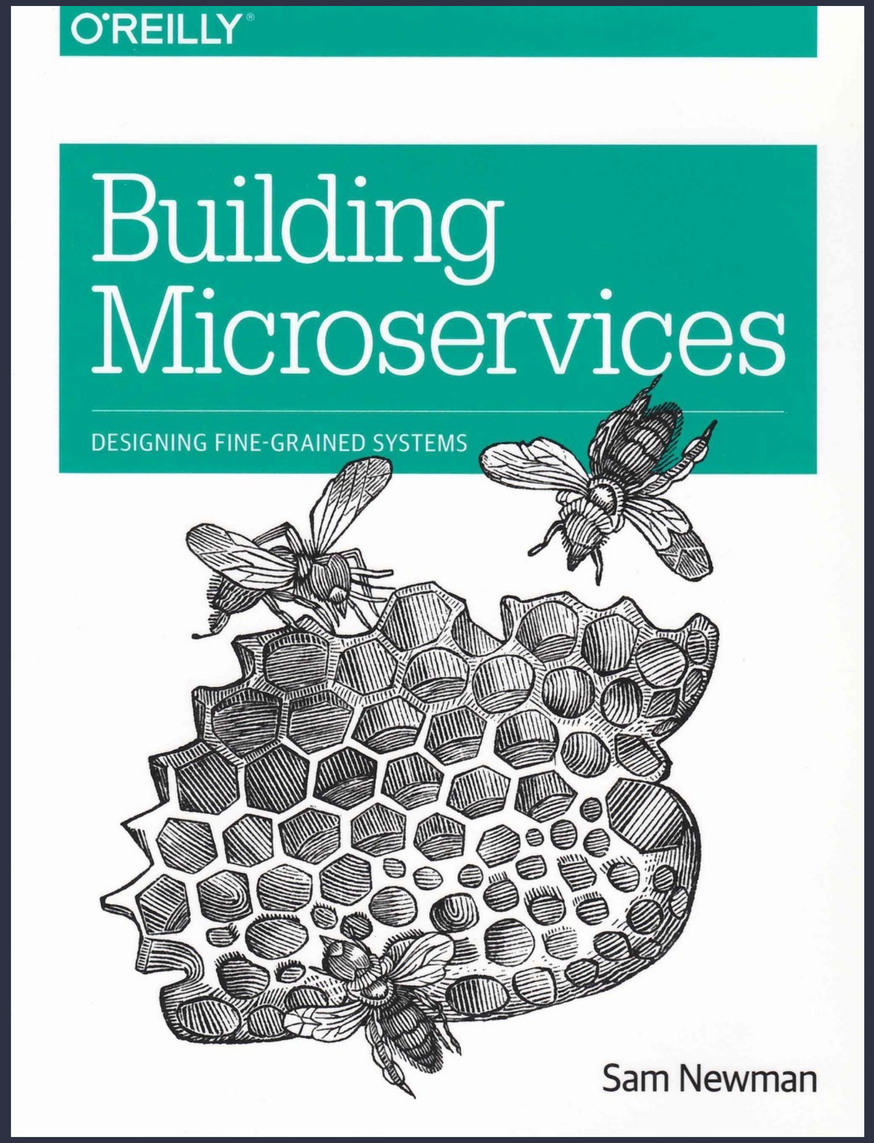
Monoliths ⟷ **Microservices**

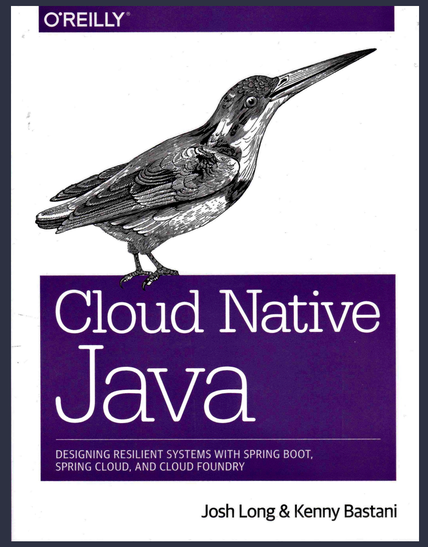
Monoliths Microservices

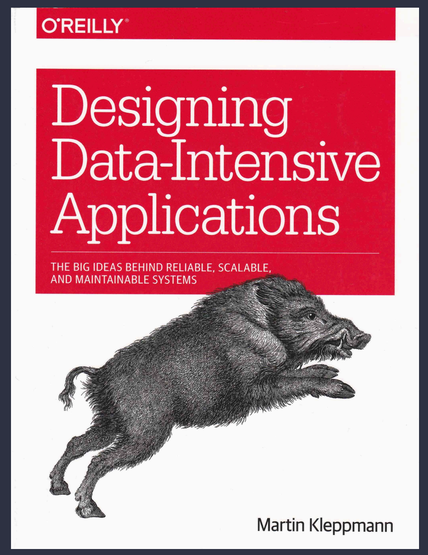

Service A
Service B
Service C
Service D


Service A
Service B
Message Bus
Service C
Service D


O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman


O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman


O'REILLY
Cloud Native Java
DESIGNING RESILIENT SYSTEMS WITH SPRING BOOT, SPRING CLOUD, AND CLOUD FOUNDRY
Josh Long & Kenny Bastani


O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann

Programming Task

Programming Task

Electricity Plans

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

Meter serial number = 4512

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am

Night Plan

Period of time

[start, end)

Meter serial number

Integer >= 0

Invoice Generator

A reusable component

Invoice, with total amount

and a list of invoice lines

One invoice line for each rate

A console application

Print generated invoice with its invoice lines on the screen

* Make the design ready for another electricity consumption plan.

Period of time = first whole 7 days of year 2020, starting in the New Year, local time

Meter serial number = 4512

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am

Night Plan

Period of time

[start, end)

Meter serial number

Integer >= 0

Invoice Generator

A reusable component

Invoice, with total amount

and a list of invoice lines

One invoice line for each rate

A console application

Print generated invoice with its invoice lines on the screen

* Make the design ready for another electricity consumption plan.

Invoices in Spring

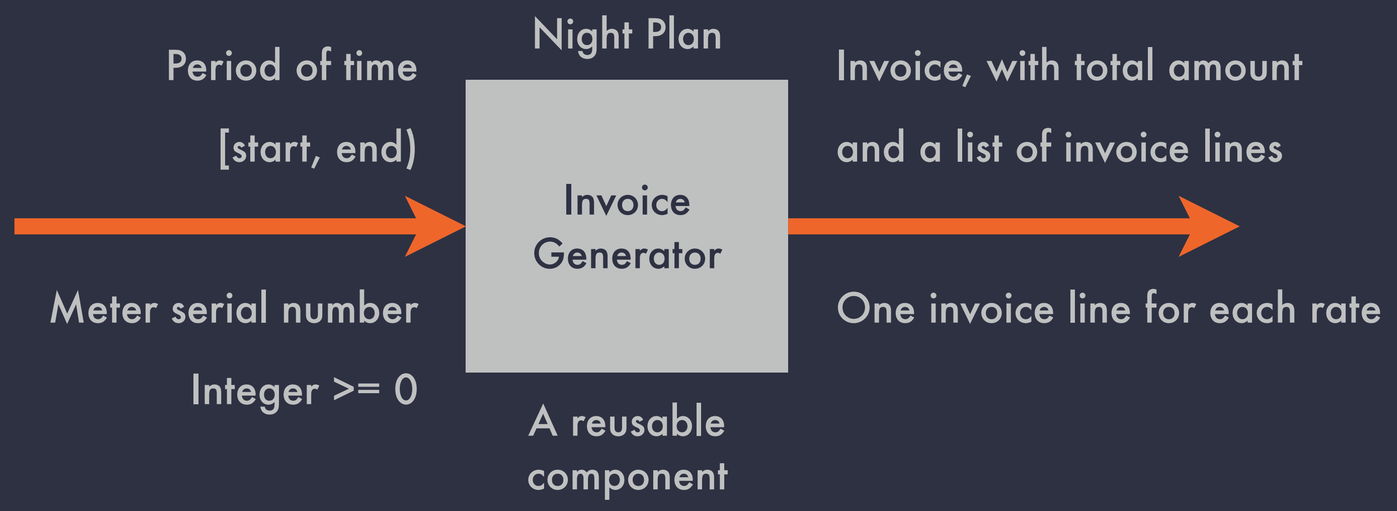
1. Invoice for a weekend plan

2. Invoice for a night plan

3. Invoice for a night plan

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am



Working day rate: 0,19288 per 1 kWh, Mon – Fri

Weekend rate: 0,17531 per 1 kWh, Sat – Sun

Weekend Plan

Day rate: 0,19288 per 1 kWh, 8:00 am – 10:59 pm

Night rate: 0,17531 per 1 kWh, 11:00 pm – 7:59 am

Night Plan

Electricity Plan

Period of time

[start, end)

Meter serial number

Integer >= 0

Invoice Generator

A reusable component

Invoice, with total amount

and a list of invoice lines

One invoice line for each rate

More Fun with Refactoring, IDE and Testing

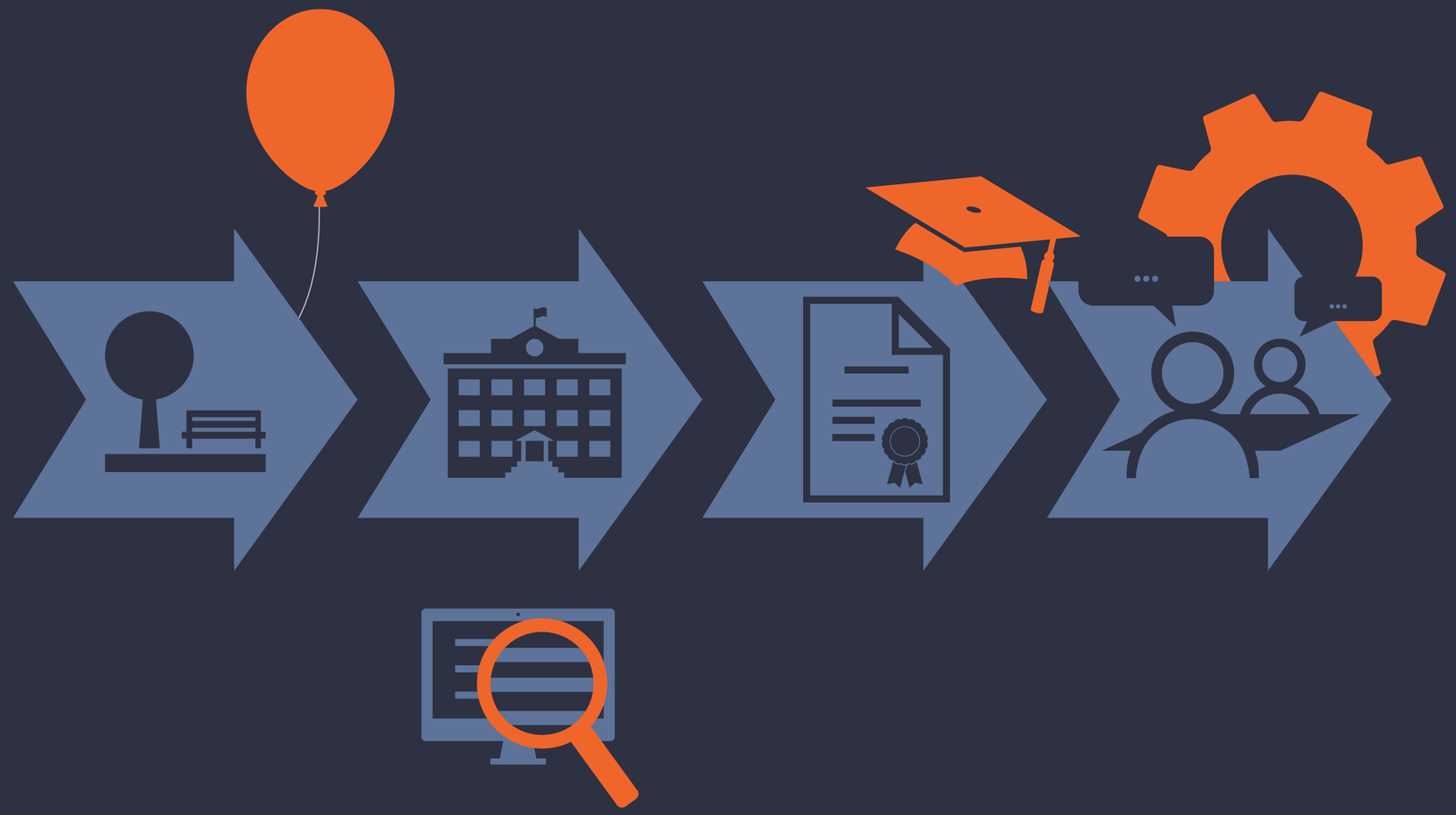
Module 6

Practicalities

Job Interviews, Sample Tasks, Learning Tips

Job Interviews

GitHub
.gitignore
README.md
docker-compose.yml
Abc



Angular

Kotlin

SQL

JavaScript

Scala

Spring

Java

React

Angular

Kotlin

SQL

JavaScript

Scala

Spring

Java

React

Databases

Networks

Principles

Version Control

Angular

Kotlin

SQL

JavaScript

Scala

Spring

Java

Methodologies

React

Databases

Networks

Principles

Version Control

Angular

Kotlin

SQL

JavaScript

Spring

Scala

Methodologies

Scrum

Java

XP

Databases

React

Networks

Version Control

Principles

Kotlin

SQL

Angular

Java

Spring

JavaScript

Scala

Unit Tests

Methodologies

Test Pyramid

Scrum

TDD

XP

React

Databases

Networks

Version Control

Principles

Kotlin

SQL

Angular

Java

Spring

JavaScript

Scala

Methodologies

Unit Tests

Scrum

Test Pyramid

XP

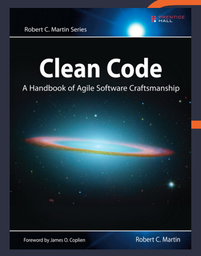
TDD

React

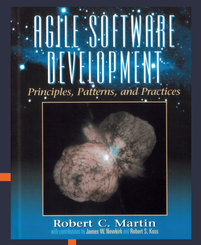
Networks

Databases

Version Control



Principles



Kotlin

SQL

Angular

Java

Spring

JavaScript

Unit Tests

Scala

Methodologies

Test Pyramid

Scrum

XP

TDD

React

Databases

Networks

Version Control

Principles

- Classes, interfaces, inheritance
- Collections
- …

JavaScript

Angular

Java

Spring

…

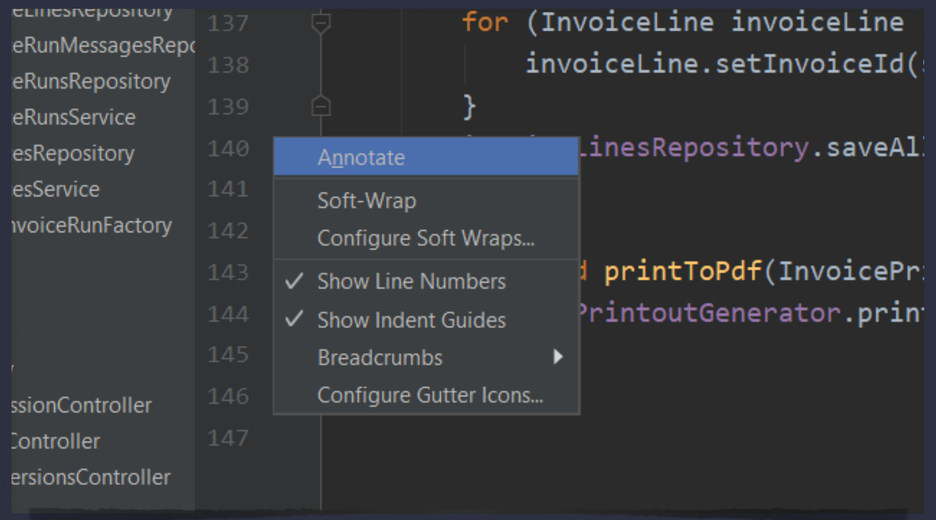
- Operations on strings, collections
- Investigations in existing code
- New unit test for existing code
- New code for existing unit test
- hackerrank.com, codewars.com, …
- Algorithms

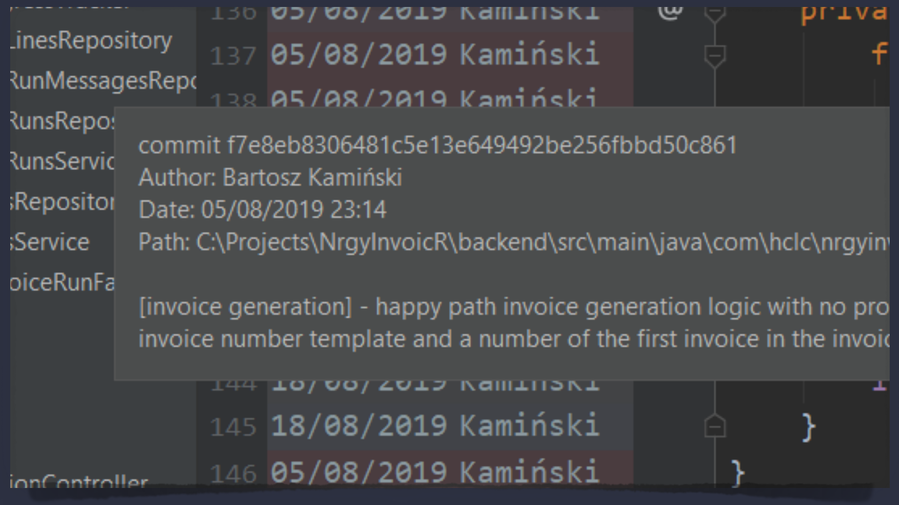
Summary

Sample Tasks

- Non-programming task
- Guided bug fixing
- Self-reliant bug fixing

- Self-reliant bug fixing
- Reproduce the problem locally
- Search by keywords in Jira
- Check Git history
- Ask for help





- Non-programming task
- Guided bug fixing
- Self-reliant bug fixing
- A new feature based on existing feature
- A new feature

Learning Tips

- Attend user groups meetings
- Attend workshops
- Learn with a friend
- Pair program at work
- Attend conferences
- Watch talks on YouTube
- Read books
- Use google (judiciously)
- Practice a lot