System Integration F2020

Exercises

Henrik Kramselund Jereminsen
hlk@zencurity.com

February 8, 2021

Contents

Preface

This material is prepared for use in *System Integration F2020* and was prepared by Henrik Lund Kramshoej, Zencurity Aps. It describes the setup and applications for trainings and workshops where hands-on exercises are needed.

Further a presentation is used which is available as PDF from kramse@Github
Look for system-integration-exercisesin the repo security-courses.

These exercises are expected to be performed in a training setting with network connected systems. The exercises use a number of tools which can be copied and reused after training. A lot is described about setting up your workstation in the repo

https://github.com/kramse/kramse-labs

Prerequisites

This material expect that participants have a working knowledge of internet from a user perspective. Basic concepts such as web site addresses, IP-addresses and email should be known as well.

Have fun and learn

Exercise content

Most exercises follow the same procedure and has the following content:

- **Objective:** What is the exercise about, the objective
- **Purpose:** What is to be the expected outcome and goal of doing this exercise
- **Suggested method:** suggest a way to get started
- **Hints:** one or more hints and tips or even description how to do the actual exercises
- **Solution:** one possible solution is specified
- **Discussion:** Further things to note about the exercises, things to remember and discuss

Please note that the method and contents are similar to real life scenarios and does not detail every step of doing the exercises. Entering commands directly from a book only teaches typing, while the exercises are designed to help you become able to learn and actually research solutions.

Exercise 1

Date Formats 15 min

Objective:
See an example of time parsing, and realize how difficult time can be in system integration.

Purpose:
System integration often works with different representations of the same data. Time and dates are one aspect we often meet. Realize how complex it is.

Suggested method:
Visit the web pages of an existing tool, Logstash we will use througout the course and a standard for time and dates.

Write down todays date on a piece of paper, each one does their own.

Then lookup ISO 8601
`https://en.wikipedia.org/wiki/ISO_8601`

I recommend looking at a specific system, used for processing computer logs: Logstash
`https://www.elastic.co/guide/en/logstash/current/plugins-filters-date.html`

Hints:
When you receive a date there are so many formats, that you need to be very specific how to interpret it.

Parsing dates is a complex task, best left for existing frameworks and functions.

If you decide to parse dates using your own code, then centralize it - so you can update it when you find bugs.

Solution:
When you have a logstash reading a date, or via a Grok debugger on the web

Discussion:
Make sure to visit the web page:
`https://infiniteundo.com/post/25326999628/falsehoods-programmers-believe-about-time`

Did you realize how complex time and computers are?

Then consider this software bug:

”No, you’re not crazy. Open Office can’t print on Tuesdays.”
https://bugs.launchpad.net/ubuntu/+source/file/+bug/248619

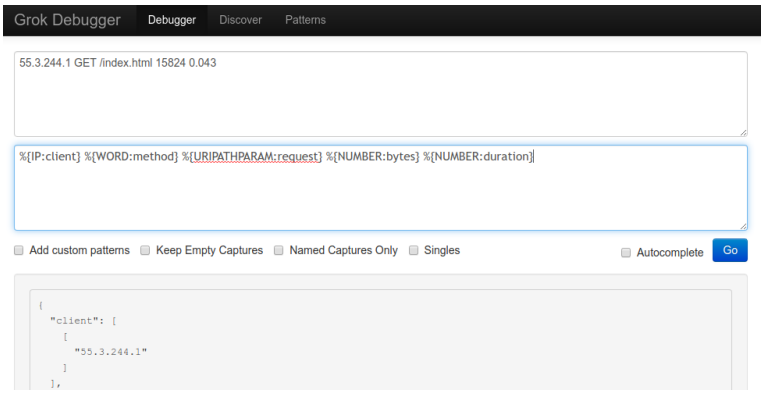
Linked from
https://www.reddit.com/r/linux/comments/9hdam/no_youre_not_crazy_open_office_cant_print_on/

Because a command `file` has an error in parsing data, files with PostScript data - print jobs with the text Tue - are interpreted as being Erlang files instead. This breaks the printing, on Tue(sdays).

We will go through this bug in detail together.

Exercise 2

Grok Debugger 15 min



Objective:
Try parsing dates using an existing system.

Purpose:
See how existing systems can support advanced parsing, without programming.

Suggested method:
Go to the web application Grok Debugger:
`https://grokdebug.herokuapp.com/`

Try entering data into the input field, and a parsing expression in the Pattern field.

Try the data from `https://www.elastic.co/guide/en/kibana/current/xpack-grokdebugger.html`

Hints:
The expression with greedy data is nice for matching a lot of text:
`%{GREEDYDATA:message}`

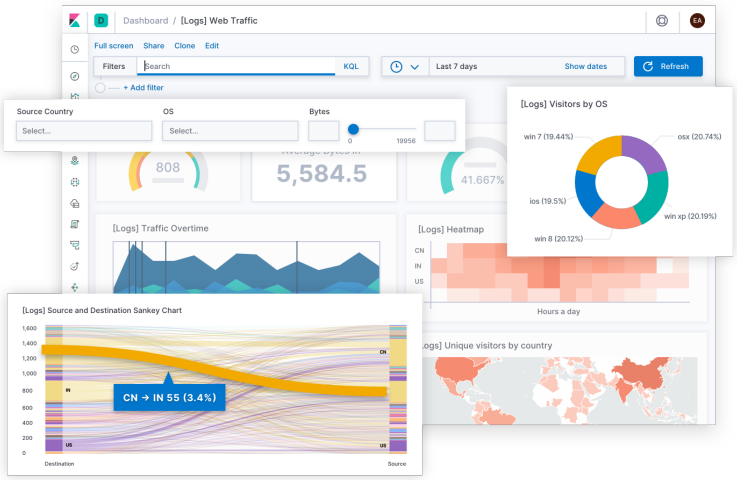
Try adding some text at the end of the input, and another part of the parsing with this.

Solution:
When you have parsed a line and seen it you are done.

Discussion:
The functionality Grok debugging is included in the tool Kibana from Elastic:
`https://www.elastic.co/guide/en/kibana/current/xpack-grokdebugger.html`

Exercise 3

Getting started with the Elastic Stack 15 min



Screenshot from `https://www.elastic.co/kibana`

Objective:
Get ready to start using Elasticsearch, read - but dont install.

Purpose:
We need some tools to demonstrate integration. Elasticsearch is a search engine and ocument store used in a lot of different systems, allowing cross application integration.

Suggested method:
Visit the web page for *Getting started with the Elastic Stack* :
`https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html`

Read about the tools, and the steps needed for manual installation.

You dont need to install the tools currently, I recommend using Debian and Ansible for bringing up Elasticsearch. You are of course welcome to install, or try the Docker method.

Hints:
Elasticsearch is the name of the search engine and document store. Today Elastic Stack contains lots of different parts.

We will focus on these parts:

- Elasticsearch - the core engine
- Logstash - a tool for parsing logs and other data.
  https://www.elastic.co/logstash

  > "Logstash dynamically ingests, transforms, and ships your data regardless of format or complexity. Derive structure from unstructured data with grok, decipher geo coordinates from IP addresses, anonymize or exclude sensitive fields, and ease overall processing."

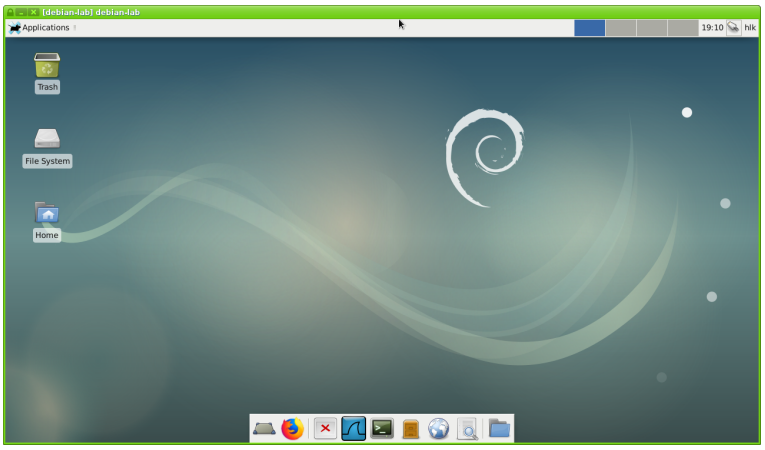
- Kibana - a web application for accessing and working with data in Elasticsearch
  https://www.elastic.co/kibana

Solution:
When you have browsed the page you are done.

Discussion:
You can read more about Elasticsearch at the wikipedia page:
https://en.wikipedia.org/wiki/Elasticsearch

Exercise 4

Check your Debian VM 10 min



Objective:
Make sure your virtual Debian machine is in working order. We need a Debian 10 Linux for running a few extra tools during the course.

This is a bonus exercise - only one Debian is needed per team.

Purpose:
If your VM is not installed and updated we will run into trouble later.

Suggested method:
Go to https://github.com/kramse/kramse-labs/ Read the instructions for the setup of a Debian VM.

Solution:
When you have a updated virtualisation software and Debian Linux, then we are good. Create a snapshot of the server, so you can return to this, in case it breaks later.

Discussion:
Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Even Microsoft has made their cloud Linux friendly, and post articles about creating Linux applications:
`https://docs.microsoft.com/en-us/azure/security/develop/`

Exercise 5

Use Ansible to install Elastic Stack

Objective:
Run Elasticsearch

Purpose:
See an example tool used for many integration projects, Elasticsearch from the Elastic Stack

Suggested method:
We will run Elasticsearch, either using the method from:
`https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html`

or by the method described below using Ansible - your choice.

Ansible used below is a configuration management tool `https://www.ansible.com/`

I try to test my playbooks using both Ubuntu and Debian Linux, but Debian is the main target for this training.

First make sure your system is updated, as root run:

```
apt-get update && apt-get -y upgrade && apt-get -y dist-upgrade
```

You should reboot if the kernel is upgraded :-)

Second make sure your system has ansible and my playbooks: (as root run)

```
apt -y install ansible git
git clone https://github.com/kramse/kramse-labs
```

We will run the playbooks locally, while a normal Ansible setup would use SSH to connect to the remote node.

Then it should be easy to run Ansible playbooks, like this: (again as root, most packet sniffing things will need root too later)

```
cd kramse-labs/suricatazeek
ansible-playbook -v 1-dependencies.yml 2-suricatazeek.yml 3-elasticstack.yml
```

Note: I keep these playbooks flat and simple, but you should investigate Ansible roles for real deployments.

If I update these, it might be necessary to update your copy of the playbooks. Run this while you are in the cloned repository:

```
git pull
```

Note: usually I would recommend running git clone as your personal user, and then use sudo command to run some commands as root. In a training environment it is OK if you want to run everything as root. Just beware.

Note: these instructions are originally from the course
Go to `https://github.com/kramse/kramse-labs/tree/master/suricatazeek`

Hints:
Ansible is great for automating stuff, so by running the playbooks we can get a whole lot of programs installed, files modified - avoiding the Vi editor ☺

Example playbook content

```
apt:
      name: " packages "
    vars:
      packages:
        - nmap
        - curl
        - iperf
        ...
```

Solution:
When you have a updated VM and Ansible running, then we are good.

Discussion:
Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Exercise 6

Run Nginx as a load balancer

Objective:
Run Nginx in a load balancing configuration.

Purpose:
See an example load balancing tool used for many integration projects, Nginx

Suggested method:
Running Nginx as a load balancer does not require a lot of configuration.

First goal: Make Nginx listen on two ports by changing the default configuration.

- Start by installing Nginx in your Debian, see it works - open localhost port 80 in browser
  `apt install nginx`
- Copy the configuration file! Keep this backup,
  `cd /etc/nginx/;cp nginx.conf nginx.conf.orig`
- Add / copy the section for the port 80 server, see below
- Change *sites* to use port 81 and port 82

Creating a new site, based on the default site found on Nginx in Debian:

```
root@debian-lab:/etc/nginx# cd /etc/nginx/sites-enabled/
root@debian-lab:/etc/nginx/sites-enabled# cp default default2
root@debian-lab:/etc/nginx/sites-enabled# cd /var/www/
root@debian-lab:/var/www# cp -r html html2
```

Then edit files `default` to use port 81/tcp and `default2` to use port 82/tcp

- also make sure `default2` uses `root /var/www/html2`

Configuration changes made.

These are the changes you should make:

```
root@debian-lab:/etc/nginx/sites-enabled# diff default default2
22,23c22,23
<         listen 81 default_server;
<         listen [::]:81 default_server;
```

```
---
>         listen 82 default_server;
>         listen [::]:82 default_server;
41c41
<         root /var/www/html;
---
>         root /var/www/html2;
```

Config test and restart of Nginx can be done using stop and start commands:

```
root@debian-lab:/var/www# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@debian-lab:/var/www# service nginx stop
root@debian-lab:/var/www# service nginx start
```

You can now visit `http://127.0.0.1:81` and `http://127.0.0.1:82` - which show the same text, but you can change the files in `/var/www/html` and `/var/www/html2`

NOTE: also verify that port 80 does *not work* anymore!

Adding the loadbalancer in nginx.conf.

We can now add the two *servers* running into a single loadbalancer with a little configuration:

Add this into `/etc/nginx/nginx.conf` - inside the section `http { ... }`

```
upstream myapp1 {
        server localhost:81;
        server localhost:82;
}

server {
        listen 80;

        location / {
                proxy_pass http://myapp1;
        }
}
```

And test using `http://127.0.0.1:80`

Hints:
Make changes to the two sets of HTML files

```
root@debian-lab:~# cd /var/www/
root@debian-lab:/var/www# diff html
```



```
html/  html2/
root@debian-lab:/var/www# diff html/index.nginx-debian.html html2/index.nginx-debian.html
4c4
< <title>Welcome to nginx!</title>
---
> <title>Welcome to nginx2!</title>
14c14
< <h1>Welcome to nginx!</h1>
---
> <h1>Welcome to nginx2!</h1>
```

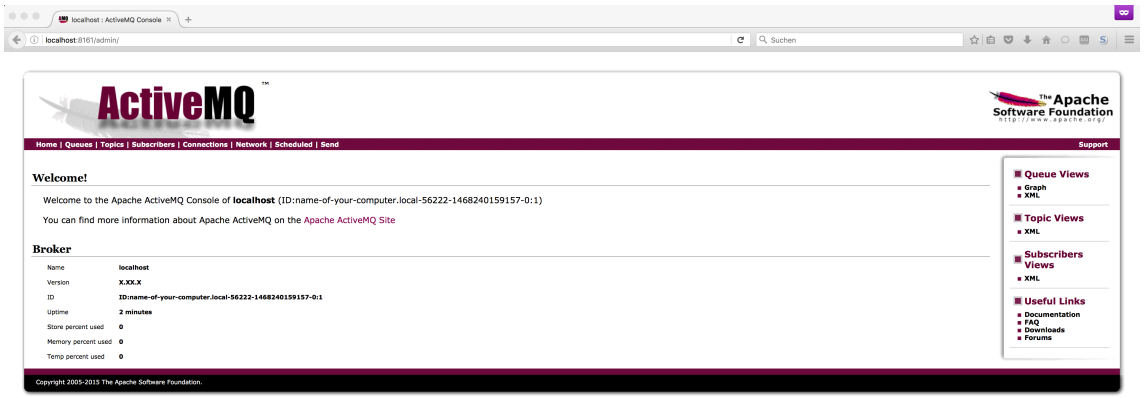
When reloading the page a few times it will switch between the two versions

Solution:
When you have Nginx running load balanced, then we are good.

Discussion:
Nginx is one of the most popular load balancers and web servers. Many sites use it for processing HTTPS/TLS before reaching application servers.

Exercise 7

Runing ActiveMQ



Objective:
Try running ActiveMQ manually - outside of Camel

Purpose:
System integration often works with JMS and ActiveMQ implements this. We can run ActiveMQ as part of Camel, but lets try running it alone.

Suggested method:
Visit the web page `https://activemq.apache.org/getting-started`, read and also follow download link.

I used `apache-activemq-5.15.11-bin.tar.gz`

Follow the instructions for getting ActiveMQ up and running on your Debian server.

Best course of actions is to use a root user and directory such as `/opt` - something like this:

```
cd /opt
tar zxvf /directory/where/you/downloaded/
mkdir data;chmod a+rwx data/
./bin/activemq console
```

Hints:
ActiveMQ is also available as a package in Debian - try searching with `apt search activemq` and can be installed with `apt install activemq`

Note: when installing the package maintainers version the configuration files are often found in the directories below `/etc` while running an unpacked tgz from the project they are close to the software. YMMV.

It is not recommended to use the Debian package for this exercise, but for production use it would be.

In this case there is a configuration file available, but not activated, try this:
`sudo ln -s /etc/activemq/instances-available/main /etc/activemq/instances-enabled/main`

When changing configuration files, if using the Debian package, you can get debug info: `/etc/init.d/activemq console main`

One problem I observed was the directory missing, which can be created using this:

```
# mkdir -p  /var/lib/activemq/data/
# chmod a+rwx  /var/lib/activemq/data/
```

afterwards when programs drop files, we can check the settings, ownership and tighten this.

Also this configuration does NOT start the web console!

Solution:
When you have a running ActiveMQ and can see the web administration you are done.

If you want to investigate further get the demos up and running:
`https://activemq.apache.org/web-samples`

Discussion:
What are the benefits of running ActiveMQ from Debian package vs running it from project binaries directly?

What version does your application need? How do you guarantee this?

Exercise 8

Runing RabbitMQ with Python

Objective:
Try running RabbitMQ with a few Python programs.

Purpose:
RabbitMQ is an alternative to ActiveMQ.

Suggested method:
Use the RabbitMQ tutorial to send a message. Use the tutorial:
`https://www.rabbitmq.com/tutorials/tutorial-one-python.html`

First you would install the server and suporting library - Pika:

```
apt install rabbitmq-server python-pika
```


Check status:

```
# service rabbitmq-server status
  rabbitmq-server.service - RabbitMQ Messaging Server
    Loaded: loaded (/lib/systemd/system/rabbitmq-server.service; enabled; vendor
    Active: active (running) since Sun 2020-03-08 13:57:38 CET; 36s ago
 Main PID: 1663 (beam.smp)
    Status: "Initialized"
     Tasks: 87 (limit: 2386)
    Memory: 77.9M
    CGroup: /system.slice/rabbitmq-server.service
            1659 /bin/sh /usr/sbin/rabbitmq-server
            1663 /usr/lib/erlang/erts-10.2.4/bin/beam.smp -W w -A 64 -MBas agef
            1898 erl_child_setup 65536
            1917 inet_gethost 4
            1918 inet_gethost 4

Mar 08 13:57:34 elastilab systemd[1]: Starting RabbitMQ Messaging Server...
Mar 08 13:57:38 elastilab systemd[1]: rabbitmq-server.service: Supervising proce
Mar 08 13:57:38 elastilab systemd[1]: Started RabbitMQ Messaging Server.
Mar 08 13:57:38 elastilab systemd[1]: rabbitmq-server.service: Supervising proce
root@elastilab:~#
```

Sender:

```
#!/usr/bin/env python
import pika
connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
channel = connection.channel()
channel.queue_declare(queue='hello')
channel.basic_publish(exchange='', routing_key='hello', body='Hello World!')

print(" [x] Sent 'Hello World!'")

connection.close()
```

Receiver:

```
#!/usr/bin/env python
import pika
connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='localhost'))
channel = connection.channel()
channel.queue_declare(queue='hello')
def callback(ch, method, properties, body):
    print(" [x] Received %r" % body)

#channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)
# With apt package python-pika 0.11.0-4, this works:
channel.basic_consume(callback, 'hello', no_ack=True)

print(' [*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
```


Hints:
Running receiver would look like this:

```
kramse@elastilab:~/projects/rabbit$ ./recv.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received 'Hello World!'
 [x] Received 'Hello World!'
 [x] Received 'Hello World!'
```

Solution:
When you have a running RabbitMQ with the Python programs working you are done.

Discussion:
RabbitMQ supports many libraries and languages, check out the list on: `https://www.rabbitmq.com/devtools.html`

What if you want to connect ActiveMQ and RabbitMQ?

Both suppoprt AMQP 1.0 - so this might be a way to support this.

Exercise 9

Runing OpenJDK for Camel

Objective:
Install OpenJDK 8 from AdoptOpenJDK on your Debian server.

Purpose:
Get OpenJDK 8 installed for the exercises to work.

Suggested method:
First clone the instructors camelinaction2 repository

```
git clone https://github.com/kramse/camelinaction2.git
```

This will download a version of the book files, where pom.xml is updated to include modules needed for them to work. Use this directory when running mvn commands from the book.

Then use the instructions from this article, to install OpenJDK 8
`https://linuxize.com/post/install-java-on-debian-10`

Then make this your default java - as root:

```
# cd /usr/bin
# mv java java.orig        //  Make sure we can go back to OpenJDK 11 later
# ln -s /usr/lib/jvm/adoptopenjdk-8-hotspot-amd64/bin/java java
```

This saves the old Java link, and creates a new link to the just installed OpenJDK 11.

Hints:
If the changes made work, you can run the OpenJDK 8:

```
  hlk@debian-lab:~$ java -version
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (AdoptOpenJDK)(build 1.8.0_242-b08)
OpenJDK 64-Bit Server VM (AdoptOpenJDK)(build 25.242-b08, mixed mode)
```

Solution:
When your java version command gives the 1.8 it works.

Discussion:
The book examples use spring and JAXB and things that aren't available in the newer Java JDK versions.

Also the pom.xml for Maven needs references updated.

Exercise 10

Run PostgreSQL

New to PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to install and use PostgreSQL through the official documentation. The PostgreSQL community provides many helpful places to become familiar with the technology, discover how it works, and find career opportunities. Reach out to the community here.

Objective:
Try a real SQL RDBMS.

Purpose:
Relational databases are used around the world for storing production data. When doing system integration projects we will often need to read or store data in databases, so a minimum of knowledge about these are needed.

Suggested method:
First visit `https://en.wikipedia.org/wiki/Relational_database` and read about relational database systems RDBMS.

Visit the home page of PostgreSQL `https://www.postgresql.org/` and read a little about this project. How mature is this, what version is current, would you run this in production?

Then go to the tutorials listed on: `https://www.postgresqltutorial.com/`

Perform the following as a minimum:

- Section 1. Getting Started with PostgreSQL
  See note about Debian below. I recommend running it on Debian
- Section 2. Querying Data
- Section 3. Filtering Data
- Section 13. Managing Databases, parts Create, Alter, Rename, Drop, Copy

Hints:
Note: Debian already has PostgreSQL in the package system, it is recommended to use the version "Included in distribution"
`https://www.postgresql.org/download/linux/debian/`

```
apt-get install postgresql-11
```

Solution:
When you have installed a PostgreSQL system and can do queries against a sample database you are done. I recommend familiarizing yourself with the basic SQL statement select with a where clause.

Discussion:
Who would choose PostgreSQL, and who would use Microsoft SQL?

Why choose Oracle SQL?

Exercise 11

Why go to SOA 45 min

Objective:
Consider why SOA is a good idea.

After reading chapters 1-5 in the SOA book

Purpose:
Think about when to use SOA, and when other models might be sufficient.

Suggested method:
Open your favourite presentation program.

Create a one-page, and one-page only, presentation about SOA in *Your Company*

Hints:
Consider this is a presentation for the management and CEO of a company. The company might have various challenges which SOA can help with. Distill the knowledge from the chapters into short sentences, bullets etc.

An idea is to split the page into fields 4-9 boxes and have small headlines ”What is”, ”Why SOA”, ”current challenges”, ”goals” and similar.

Solution:
When you have a one-page presentation that summs up why you should use SOA you are done. I would like to see them, but the main goal of this exercise is to consider SOA as a tool, and know when to use it.

Discussion:
Instead of SOA your company might have used industry standard tools like JSON, XML, REST, WSDL etc. but is it enough to ensure future interoperability?

Notes: The above reference to the SOA book, is the book:
Service-Oriented Architecture: Analysis and Design for Services and Microservices, Thomas Erl, 2017 ISBN: 978-0-13-385858-7

Exercise 12

Cloud Computing Introduction 45 min

Objective:
Many businesses today use cloud services, but what are they?!

Purpose:
Have minimum knowledge about the term cloud computing.

Suggested method:
Read `https://en.wikipedia.org/wiki/Cloud_computing`

Take special notice of the relation to client–server and mainframe which has been used a lot before, and often represent the systems to integrate with cloud.

Then read a little about two cloud systems:

- Kubernetes `https://en.wikipedia.org/wiki/Kubernetes`
  and the homepage `https://kubernetes.io/`
- Microsoft Azure `https://en.wikipedia.org/wiki/Microsoft_Azure`

Hints:
Some people run K8S inside Azure, why? To allow them to move applications from their on-site / on-premise cloud into the Azure cloud on the internet easily.

Solution:
You are done, when you have read the main cloud computing wikipedia page. You should have an idea about cloud computing and the various terms Platform as a service (PaaS) and Software as a service (SaaS).

Discussion:
Other cloud systems exist, and today system integration projects often have a part with cloud integration.

Exercise 13

Cloud Deployment 45 min

Objective:
See how you would deploy an application into a cloud, using the Microsoft Azure cloud example *Develop a secure web app*

Purpose:

Suggested method:
Go to and read: `https://docs.microsoft.com/en-us/azure/security/develop/secure-web-app`

You are not expected to work through the example, but get an idea about how you deploy applications securely in the modern internet.

Hints:
Security is a huge part of running computing today, but often forgotten.

The example both describes the architecture of the applications, the security implications, and the actual deployment of an app.

Solution:
When you have read the example you are done.

Discussion:
If you want to work with cloud technologies you would often use Azure, but also other clouds like Amazon Web Services (AWS) are popular in Denmark.

Read more about AWS in wikipedia:
`https://en.wikipedia.org/wiki/Amazon_Web_Services`

Exercise 14

Download the Microservices ebook 20 min

Objective:
Download the book: *Microservices for Java Developers* by Christian Posta **Purpose:**

Suggested method:
Download the book via the link from:
`https://www.oreilly.com/programming/free/files/microservices-for-java-developers.pdf`

Hints:

Solution:
When you have a copy of the PDF you are done.

Discussion:
You are welcome to read chapter 1 of the book.

Exercise 15

Postman API Client 20 min

Objective:
Get a program capable of sending REST HTTP calls installed.

Purpose:
Debugging REST is often needed, and some tools like Elasticsearch is both configured and maintained using REST APIs.

Suggested method:
Download the app from https://www.postman.com/downloads/

Available for Windows, Mac and Linux.

Hints:
You can run the application without signing in anywhere.

Solution:
When you have performed a REST call from within this tool, you are done.

Example: use the fake site https://jsonplaceholder.typicode.com/todos/1 and other similar methods from the same (fake) REST API

If you have Elasticsearch installed and running try: http://127.0.0.1:9200

Discussion:
Multiple applications and plugins can perform similar functions. This is a standalone app.

Tools like Elasticsearch has plugins allowing decoupling of the API and plugins. Example: https://www.elastic.co/what-is/elasticsearch-monitoring and https://www.elastic.co/what-is/open-x-pack

Exercise 16

Getting started with the Elastic Stack - 60 min

Objective:
Get a working Elasticsearch, so we can do requests.

Purpose:
Elasticsearch uses REST extensively in their application.

Suggested method:
either use the *Getting started with the Elastic Stack* https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html

OR my Ansible based approach - which some already ran.

The ansible is described in exercise 5 on 8

Hints:
We dont really need a lot in the Elasticsearch database, and you can run most tasks with zero data. Graphs will not be as pretty though.

Solution:
When you have a running Elasticsearch you are done, and ready for next exercise.

The web page for the getting started show multiple sections:

- Elasticsearch - the core engine, this must be done manually or with Ansible
- Kibana - the analytics and visualization platform
- Beats - data shippers, a way to get some data into ES
- Logstash (optional) offers a large selection of plugins to help you parse, enrich, transform, and buffer data from a variety of sources

Each describes a part and are recommended reading.

Discussion:
We could have used a lot of other servers and service, which ones would you prefer?

If you have access to Azure, you can try Azure REST API Reference
https://docs.microsoft.com/en-us/rest/api/azure/

Exercise 17

Making requests to Elasticsearch - 15-75min

Objective:
Use APIs for accessing Elasticsearch data, both internal and user data.

Purpose:
Learn how to make requests to an API.

Suggested method:
Go to the list of exposed Elasticsearch REST APIs:
https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html

The Elasticsearch REST APIs are exposed using JSON over HTTP.

Select a category example, Cluster APIs, then select Nodes Info APIs. This will show URLs you can use:

```
# return just process
curl -X GET "localhost:9200/\_nodes/process?pretty"
# same as above
curl -X GET "localhost:9200/\_nodes/_all/process?pretty"

curl -X GET "localhost:9200/_nodes/plugins?pretty"

# return just jvm and process of only nodeId1 and nodeId2
curl -X GET "localhost:9200/\_nodes/nodeId1,nodeId2/jvm,process?pretty"
# same as above
curl -X GET "localhost:9200/\_nodes/nodeId1,nodeId2/info/jvm,process?pretty"
# return all the information of only nodeId1 and nodeId2
curl -X GET "localhost:9200/\_nodes/nodeId1,nodeId2/_all?pretty"
```

When you can see this works, then feel free to install X-Pack and monitoring plugins

Hints:
Pretty Results can be obtained using the pretty parameter.

> When appending ?pretty=true to any request made, the JSON returned will be pretty formatted (use it for debugging only!). Another option is to set ?format=yaml which will cause the result to be returned in the (sometimes) more readable yaml format.

Lots of tutorials exist for accessing Elasticsearch

A couple of examples:

- https://aws.amazon.com/blogs/database/elasticsearch-tutorial-a-quick-start-guide/
- https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04

Solution:
When you have seen examples of the API, understand the references with underscore, like `_nodes` and pretty printing you are done.

I recommend playing with Elasticsearch plugins and X-pack.
https://www.elastic.co/downloads/x-pack

Note: In versions 6.3 and later, X-Pack is included with the default distributions of Elastic Stack, with all free features enabled by default.

Also Kibana can be used for creating nice dashboards and become applications more or less.

Discussion:
You can also try calling the REST API from Python

Similar to what we did previously in this course:

```
#!/usr/bin/env python
import requests
r = requests.get('https://api.github.com/events')
print (r.json());
```

