SWitCH – Computing systems and networks (SCOMRED)

2019/2020

Laboratory Script – Week 06

BASH individual assessment training.

Deploying contents to servers.

Team project development.

1. Development environment and production environment

Under the point of view of software engineering, the software lifecycle is made of several stages, starting with planning, analysis and design, then development together with testing, and finally deployment, operation and maintenance. Of course then it will often start again (it's a cycle).

Different stages in the software lifecycle will require different skills, tools, and possibly, different running environments for the application being developed. Three clearly distinct running environments can be identified for the application: **development environment**, **testing environment**, and **production environment**.

1.1. Development

Most appropriately, an Integrated Development Environment (IDE) should be used when developing applications (development stage), usually the IDE encompasses a running environment for the application, and thus it can be tested while being developed.

However, the availability of a running environment together with the development environment is not guaranteed. Generally speaking, it may not be available if the **production environment** (where the application was ultimately designed to be used) is substantially different from the **development environment**.

One example that immediately may strike our mind is developing an Android or IOS application in a Windows, Linux, or OS X workstation. For this specific scenario, the most often used solutions are either a device emulator or a real device directly attached to the workstation.

When developing a Web application backend, or distributed applications in general, it might as well not be possible to run and test the application in the development environment, simply because the required running environment is too complex to be established in the development environment. Take for instance a set of different applications that are supposed to run each on a different network node, using different web servers and database servers. Even if it was possible to setup all those services in the development environment it wouldn't be a good testing environment because all applications and services would be running on the same node. A testing environment is a better solution.

1.2. Testing environments

If a running environment for the application is not available in the development environment, then an external testing environment is required, from one example above, it may be an Android or IOS emulator.

Even if a running environment for the application is available in the development environment, when developing distributed applications with interdependencies, and applications depending on special network services, a complex testing environment is required. The ultimate solution is creating a testing environment that mimics as far as possible the production environment, such a testing environment is often called sandbox.

Because it's a separate environment, testing over the sandbox will not disturb the production environment, and yet because they are very alike, once the applications are thoroughly tested, transferring them into production is pretty straight forward.

Operating systems virtualization is now becoming an alternative to create an instant testing environment. By starting several containers, each running a different application or network service, a testing environment is created with the single purpose of running once the application. Such testing environment only exists while the application is running, once the application ends running containers are stop and most often destroyed (ephemeral containers). One possible advantage of this kind of testing environment it that the environment is always the same when the application test starts.

The use of an instantly created containers based testing environment, doesn't mean a more full testing environment that better mimics the production environment, will not be required. However, in such case it will be then a clearly independent testing phase following the development.

1.3. Deployment

In several scenarios applications must be pushed into servers, this basically means uploading file. This is the case for production servers and as well for testing servers in a full testing environment similar to the production environment.

To ensure a total independence from the IDE, from development libraries, and from development frameworks, production hosts should be used run final products. Such production hosts will simply not have the development related software packages installed.

Many popular IDEs provide deployment mechanisms, most often based on SFTP (SSH File Transfer Protocol, also known as Secure File Transfer Protocol). SFTP is an extension to the SSH protocol allowing file transfers in a FTP style, though it's not related to FTP (File Transfer Protocol) neither to FTPS (FTP over SSL/TLS).

Most SSH servers support SFTP, so, if you have SSH access to a host, it's likely you have SFTP access as well, by using the same credentials.

Compatibility between the development host and the production host is required. If it's an interpreted application, things are simple because the ASCII encoding is pretty much universal, thus text is text everywhere[1]. Nevertheless the interpreters' versions must be compatible, usually backward compatibility is assured.

If it's a compiled application, binary compatibility is required. Java compiled programs, as we know, run on a special environment called Java Runtime Environment (JRE), it includes the Java Virtual Machine (JVM). For each operating system, a specific JRE implementation is required, yet the JRE is always the same, thus it behaves as an abstraction layer, meaning, the same compiled Java application will run over the JRE whatever the underlying operating system is.

For instance, a Java application may be developed and compiled in an MS-Windows system, then compiled files may be copied to a Linux system, and the application runs there with no changes. Even so, there are several Java versions, backwards binary compatibility is assured, e.g. an application developed and compiled in Java 8 will run on Java 11 runtime.

[1] Actually, text files in UNIX like systems (e.g. Linux) are slightly different from text files in Windows systems in the way text lines are ended. In DOS/Windows, text files each line ends with a Carriage Return (CR) followed by a Line Feed (LF). In UNIX text files each line ends with Line Feed (LF). In Mac text files, prior to Mac OS X, lines end with Carriage Return (CR). Nowadays Mac OS uses UNIX style (LF) line breaks.