Development Life Cycle & Docker

2019.11.29 Xianyi Cui

Agenda

- Software Development Life Cycle(SDLC)
- Environment in Development
- Docker
- Workflow Review & Optimize
- What's more

Software Development Flow

Current (.NET Framework)

1. Request new Windows machine
2. Install **Visual Studio**, **Git**
3. Install **MySQL**,**SQL Server**, **ElasticSearch**, **Redis**, **MongoDB**
4. Install **Azure SDK**, **Service Fabric SDK**
5. Pull code
6. Generate Dll(library)/Exe(executable)
7. Test executable or libary behaviour
8. Copy **binary** to others for usage/Use Azure SDK to deploy to Cloud

Current (.NET Core)

1. Request new Windows/Linux machine
2. Install **Visual Studio**, **Git**, **.NET Core SDK(Runtime included)**
3. Install **MySQL**,**SQL Server**, **ElasticSearch**, **Redis**, **MongoDB**
4. Install **Azure SDK**, **Service Fabric SDK**
5. Pull code
6. Generate Library Dll/Executable dll
7. Test executable or library behaviour
8. Copy **binary** to others for usage/Usage Azure SDK to deploy to Cloud

Current (Python)

1. Request new Windows/Linux machine
2. Install **Python**, **Git**, **VS Code**
3. Pull code
4. Test python file
5. Copy **python file** to others for usage

Software Development Life Cycle

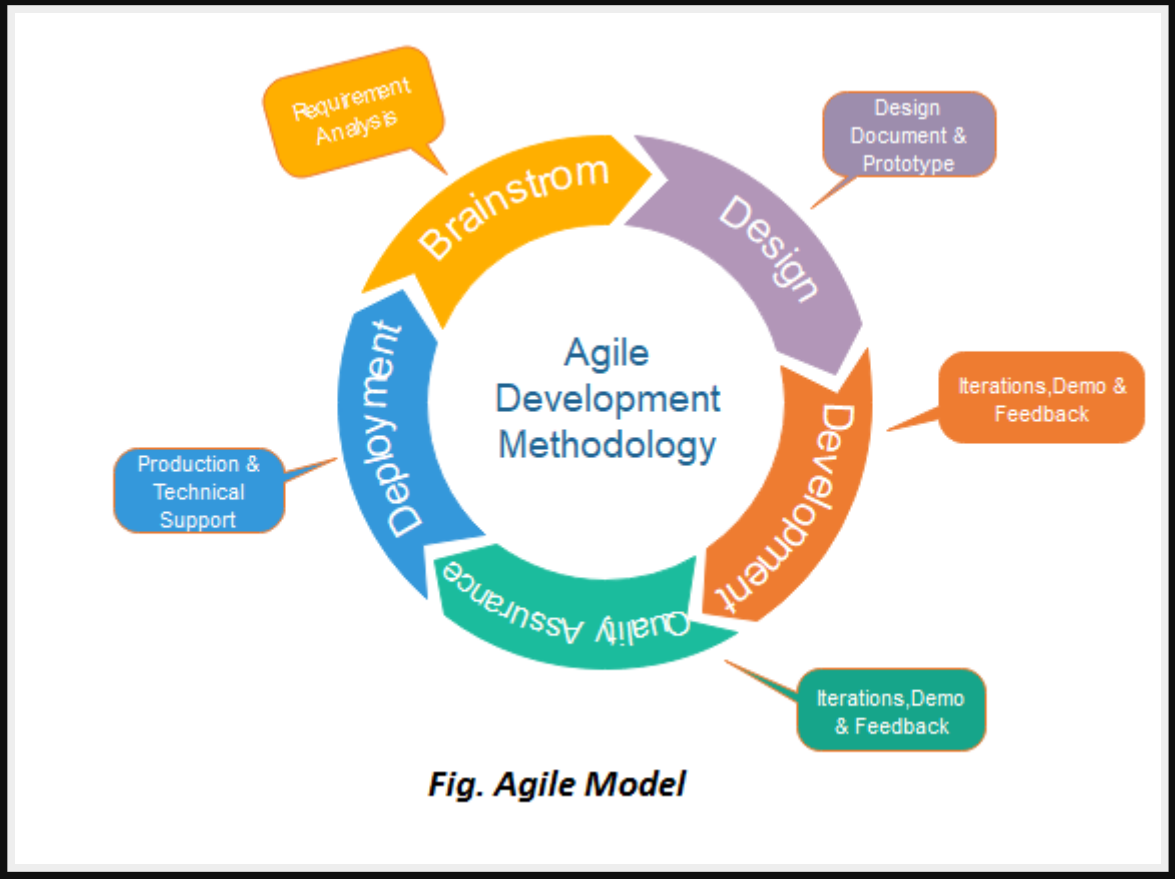


Fig. Agile Model

Focus on Developer side

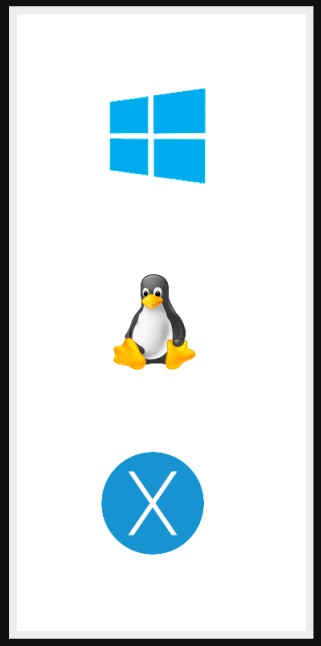
- Prepare Environment
- Coding
- Compiling
- Test (Local Run)
- Publish/Deploy
- Monitor

Focus on Developer side

- Prepare Environment
- Coding
- Compiling
- Test (Local Run)
- Publish/Deploy
- Monitor

Environment in Development

Prepare for what?



OS | Editor & IDE | **SDK & Runtime** | DB & Dependency | DevOps

OS & SDK/Runtime

Question	Answer
Can we work without **IDE/Editor**	Yes
Can we work without **OS**	No
Can we **Compile/Run** code without **SDK/Runtime**	No
Can we run our service without **DB**	Yes
Must we deploy service/manage code with git	No

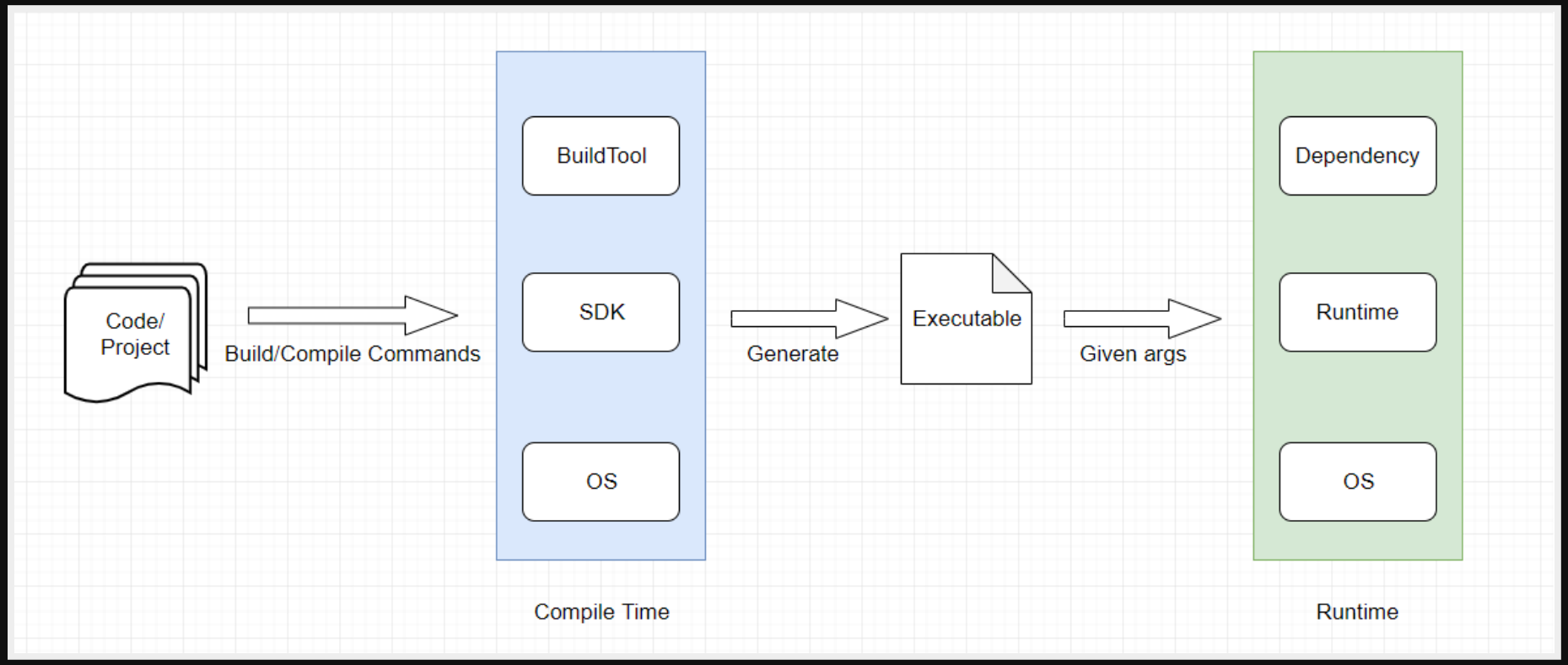
What SDK/Runtime provide

1. Compiler
2. Build tool
3. Package management
4. SDK
5. ……

All we care most is to get our code compiled on some environment and run on some target environment

All we care most is to get our code compiled on some environment and run on some target environment

Compile to Run



- **Compile time:** JDK, .NET Core SDK
- **Runtime:** JRE, .NET Core Runtime

Deep dive: Compile time

Turn **code logic** into **executable/library binary** target **specific runtime** with **dependencies & tools** under **SDK Prepared environment**

.NET Framework Compile time

Turn **c# file/project** into **xxx.dll/xxx.exe** target **.NET Framework(Windows)** with **msbuild & nuget** under **Windows installed with .NET SDK**

- BuildTool

```
MSBuild MyApp.sln /t:Rebuild /p:Configuration=Rel
```

- Compiler

```
csc -define:DEBUG -optimize -out:File2.exe *.cs
```

.NET Core Compile time

Turn **c# file/project** into **xxx.dll** target **portable .NET Core runtime(Cross platform)** with **dotnet cli & nuget** under **Linux/Windows installed with .NET Core SDK**

- BuildTool

```
dotnet build -c Release
```

- Compiler

```
csc -define:DEBUG -optimize -out:File2.exe *.cs
```

Java Compile time

Turn **java file/project** into **xxx.jar** target **java(JVM) runtime** with **maven/gradel** under **Linux/Windows installed with JDK & maven/gradel**

- BuildTool

```
mvn package
ant
gradle build
```

- Compiler

```
javac -d ./build *.java
```

Native Compile time

Turn **cpp/c file/project** into **xxx.dll/xxx.so/xxx.dylib/xxx.exe/xxx** target **win x64/x86, mac x64, linux x64, arm x86/x64** with **make/cmake/bazel** under **Linux/Mac/Windows installed with gcc & make/cmake/bazel**

- BuildTool

```
bazel / make
```

- Compiler

```
g++/gcc -o hello hello.cpp
```

Deep dive: Runtime

Make **executable/libary binary** execute on the **Runtime prepared environment** with given **arguments/command**

.NET Framework Runtime

Make **xxx.exe** execute/ref by other code on **Windows installed with .NET Framework Runtime(By default)** with **xxx.exe {args}**

```
xxx.exe arg1 arg2
```

.NET Core Runtime

Make **xxx.dll** execute/ref by other code on **Linux/Windows installed with .NET Core runtime** with **dotnet xxx.dll {args}**

```
dotnet xxx.dll arg1 arg2
```

Java Runtime

Make **xxx.jar** execute/ref by other code on **Linux/Windows installed with JRE** with **java xxx.jar {args}** command

```
java xxx.jar arg1 arg2
```

Native Runtime

Make **xxx.dll/xxx.so/xxx.dylib** ref by other code or **xxx.exe/xxx** execute on **Linux/Windows/Mac** with **xxx.exe/xxx {args}**

Look back again

- It is tedious to install various software/dependency.
- Our disk space getting smaller & we don't know where is the data.
- It is hard for us to create a new clean environment.
- How can we make sure the others' environment?

Let's think more

1. Linux vs Windows
2. Common used windows folder
3. Always think about clean environment/depenency when you write code
4. If you have depedency make it installed on target environment on **runtime**
5. Virtual machine? virtualenv(python)?
6. Basic no way. Create GHOST image?

Docker

What's Docker

- Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host.

Image & Container

Docker(Containerization)	VM(Virtualization)
Docker image	Base image(ISO/VHD)
Docker container	Virtual machine instance

- **Virtual machine:** Create single/multiple virtual machine instance with base image
- **Docker:** Run docker image and get container instance/instances

Image & Container

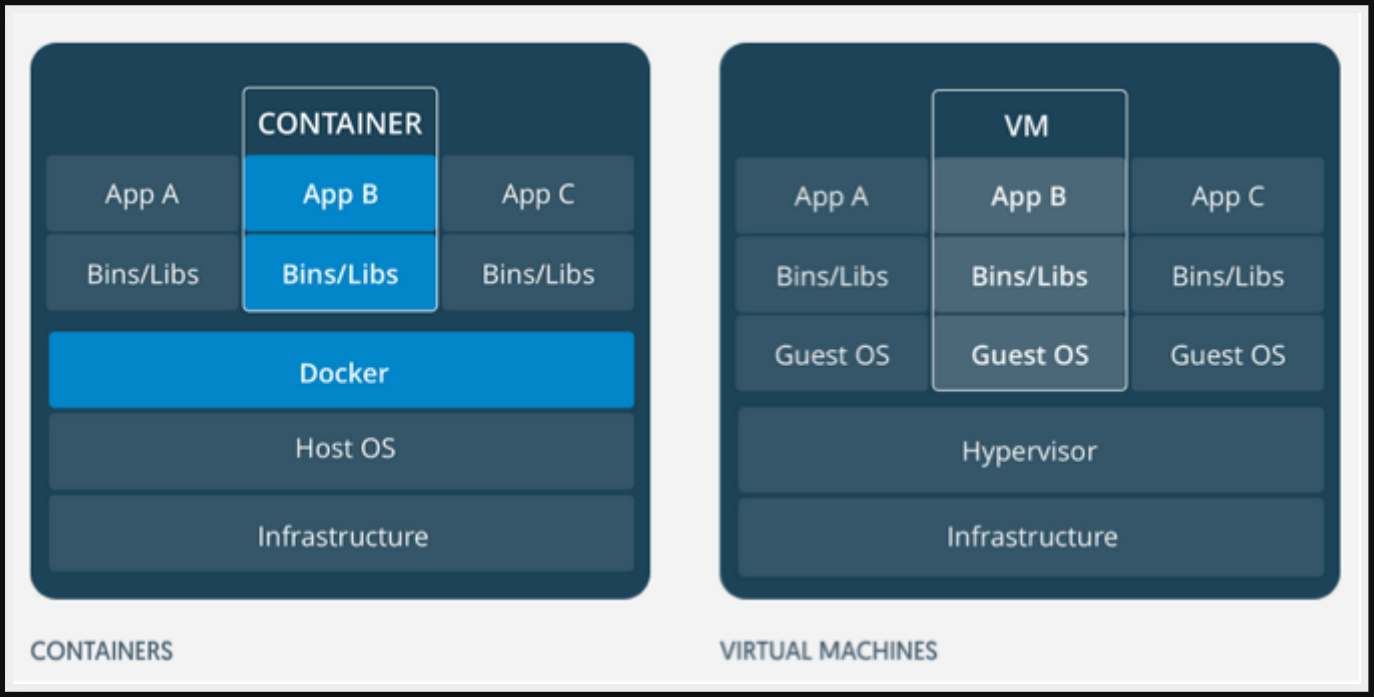


Image & Container

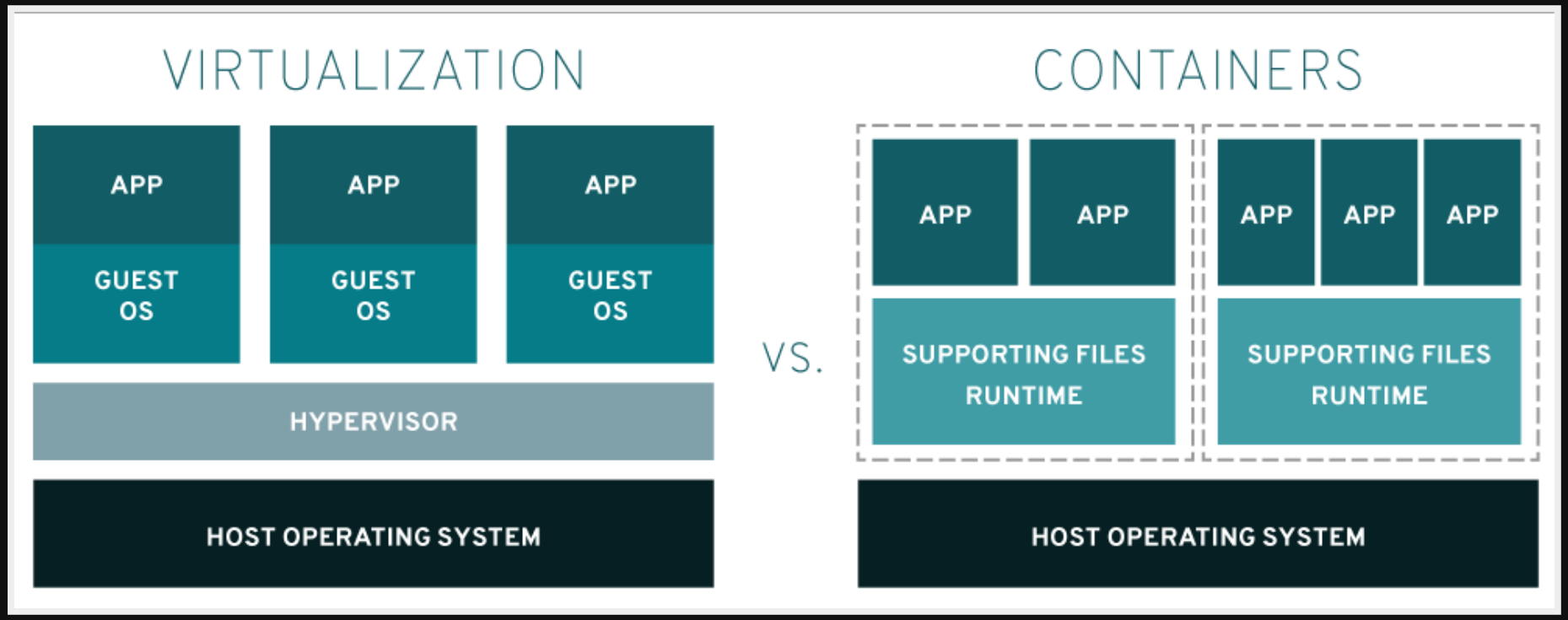
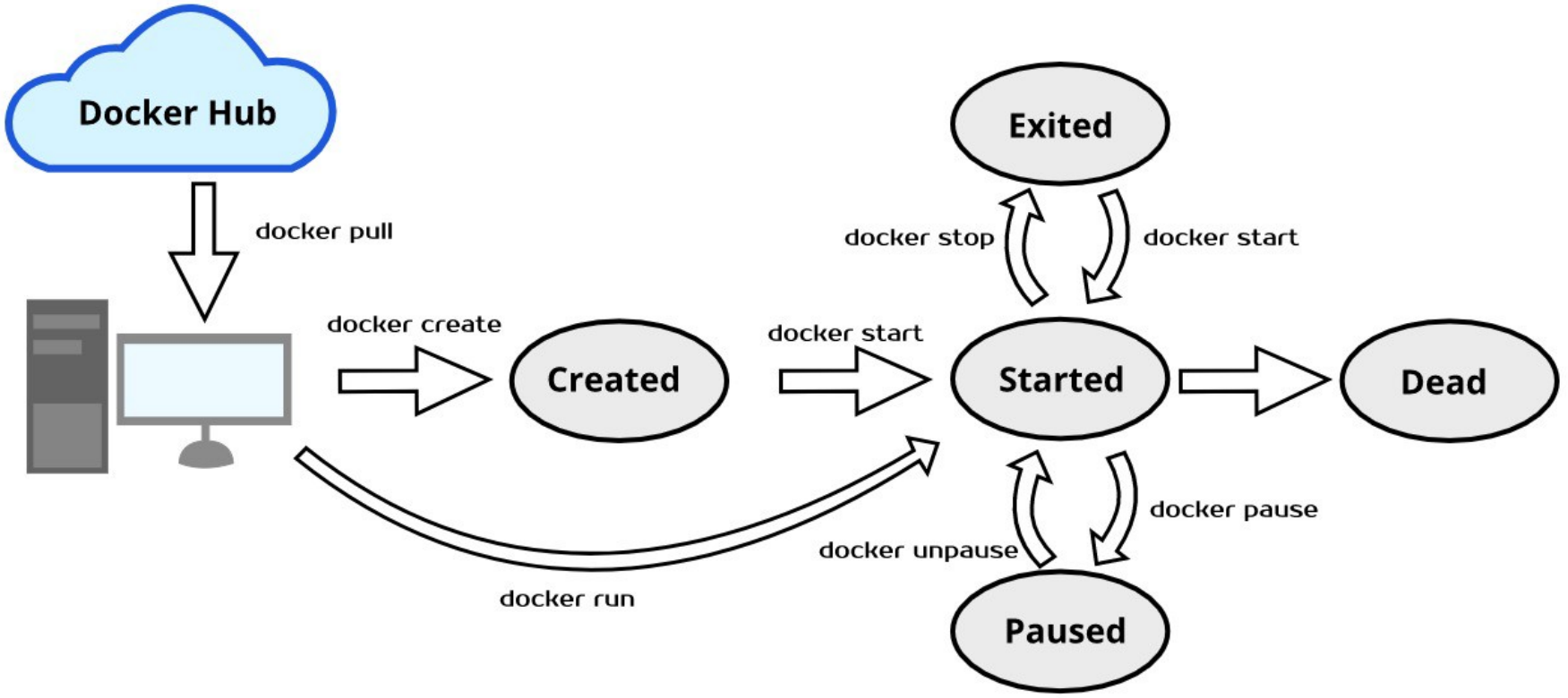
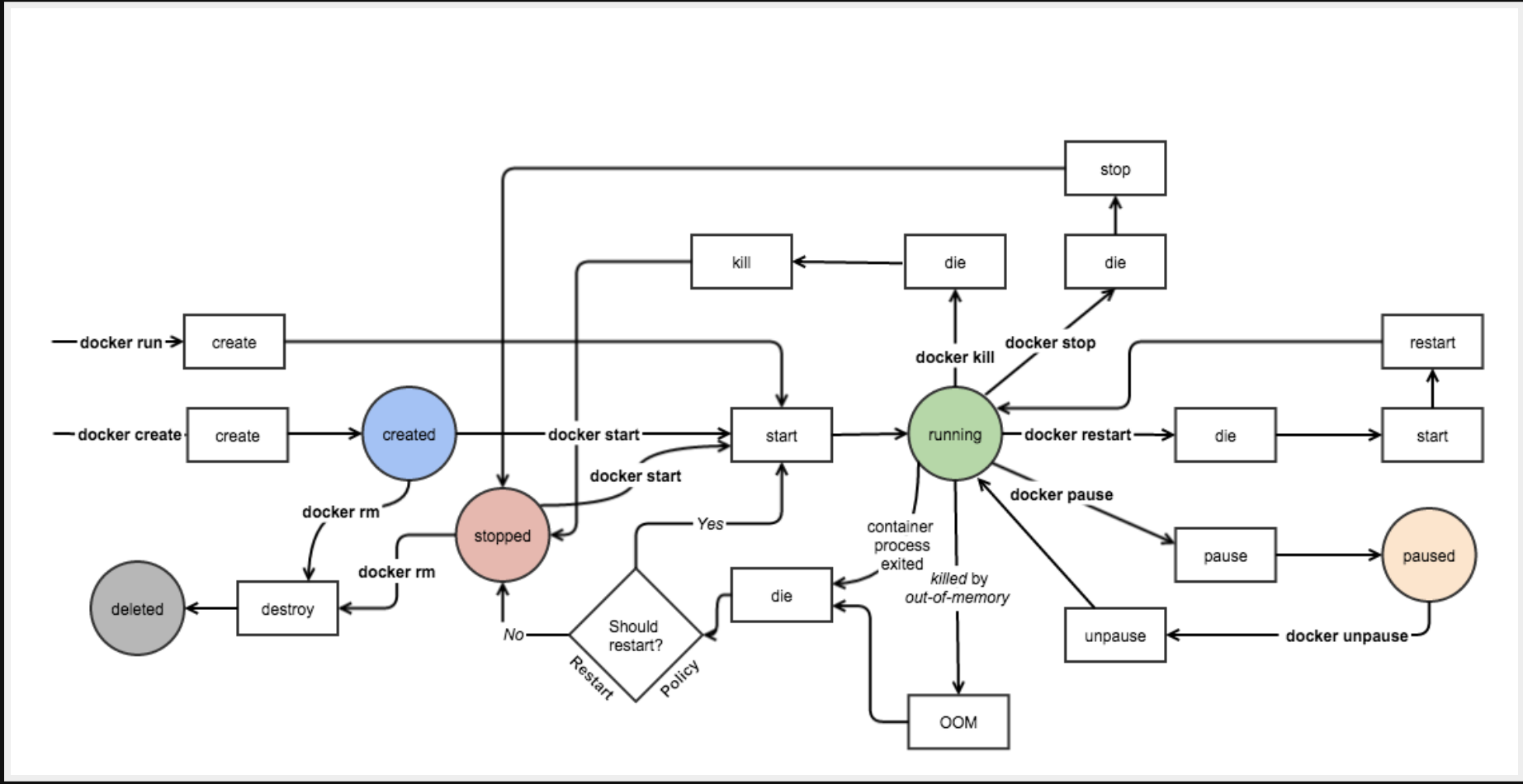


Image & Container



Docker CLI & Life cycle



What image (could) included

1. OS
2. SDK/Runtime
3. DevOps tools
4. Depenedency

How Docker Affect Compile/Run time

Item	Before	After
Env Prepare	Install SDK/Runtime on machine	docker pull && docker run
Version Select	Almost impossible	change docker image & run

Item	Before	After
Dependency Prepare	Install as much as dependency you need	docker pull && docker run
Clean installation	Registry? AppData? Program File?	docker image rm

How Docker Affect Dev LifeCycle

Item	Before	After
Product	Executable	Docker image
Reuse level	Library	Library + docker image
Target Deploy Envrionment	Well prepared	Just docker

Workflow Review & Optimize

.NET Core(Playground)

1. Use Current Windows/Linux Machine
2. Install Git, Editor, Docker.
3. Pull MySql, Redis, etc. image if you need(Runtime).
4. Pull code
5. Run sdk image && attach folder
6. Compile in SDK container
7. Run runtime image && attach folder

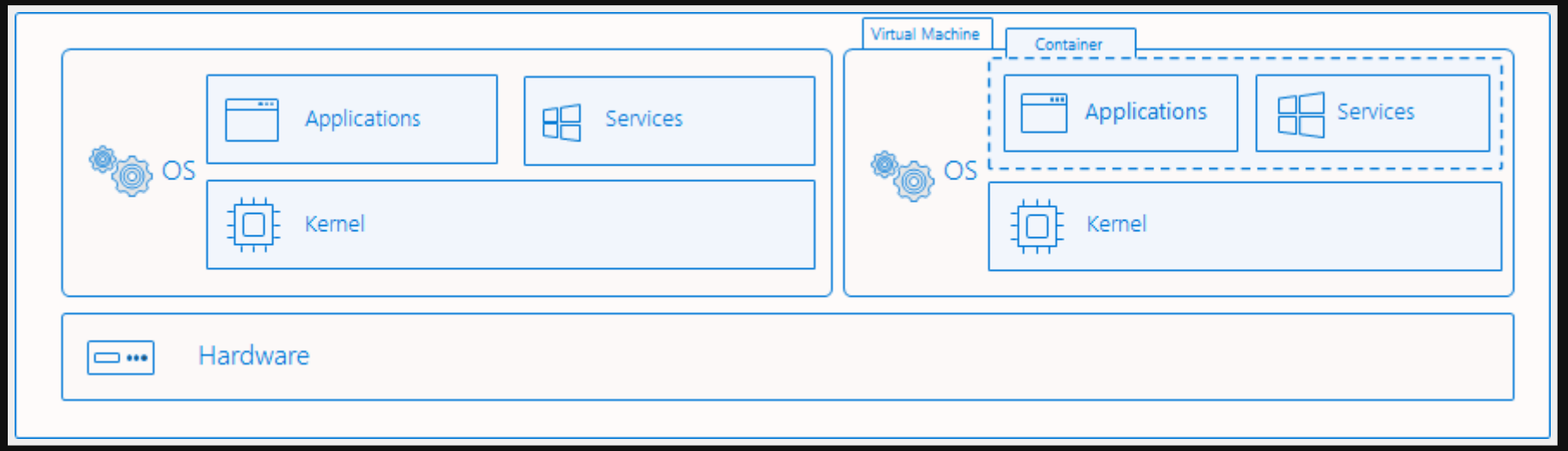
.NET Core(Product)

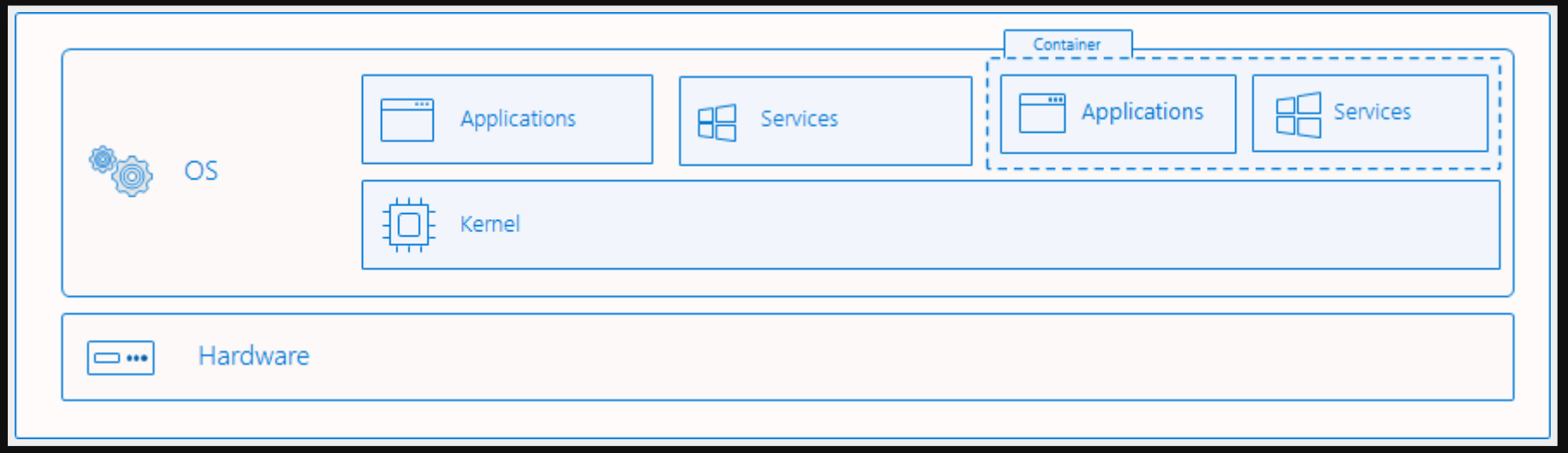
1. Use Current Windows/Linux machine
2. Install Git, Editor/IDE, Docker, SDK(For Debug)
3. Pull code
4. Write DockerFile(describe docker build)
5. Write docker-compose.yml(for dependency)
6. Docker build
7. docker run / docker-compose up to run.

Practice

What's More

Windows containerization





What about frontend development flow

What about machine learning flow

What about work flow

- Present content
- (Raw content + metadata) + host = Present
- Markdown, HTML, Web server, node, python,

What is Deploy

Make the executable program(compiled binaries/source code) runs on remote environment prepared ready

Phrases

Windows/Linux/Mac
Git/SVN/TFS
JDK/**.NET Core SDK**/.NET SDK/
NuGet/Maven/Gradle/Pip/Npm
Visual Studio/Eclipse/XCode/
Sublime/**VS Code**/Notepad++
Yeoman
gcc/javac/csc
MSBuild/**dotnet build**/Make/Bazel
JUnit/**NUnit**/**xUnit**/JMeter/Siege/Selenium .NET Framework/.NET Core/JRE/Python/Nodejs

Reference

- SDLC: https://en.wikipedia.org/wiki/Software_development_process
- SDLC: https://stackify.com/what-is-sdlc/
- Agile vs DevOps: https://decideconsulting.com/sdlc-comparision-agile-vs-devops/
- Agile vs DevOps: https://www.guru99.com/agile-vs-devops.html
- Devops lifecycle: https://medium.com/edureka/devops-lifecycle-8412a213a654
- Software Development lifecycle: https://medium.com/@jilvanpinheiro/software-development-life-cycle-sdlc-phases-40d46afbe384

Reference(Container)

- Isolation Modes: https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/hyperv-container

Reference(Docker)

- Docker life cycle: https://medium.com/@nagarwal/lifecycle-of-docker-container-d2da9f85959
- Docker & VM: https://medium.com/faun/introduction-to-docker-life-cycle-3bf3aeba883
- Architecture: https://docs.docker.com/engine/docker-overview/#docker-architecture
- Container (zh-cn): https://zhuanlan.zhihu.com/p/39155341

Reference(Build tool)

- Maven: https://maven.apache.org
- Gradel: https://gradle.org/
- Ant: https://ant.apache.org/

Reference(Tools)

- Markdown Guide: https://www.markdownguide.org/basic-syntax/
- Reveal.js: https://github.com/hakimel/reveal.js/

Thanks