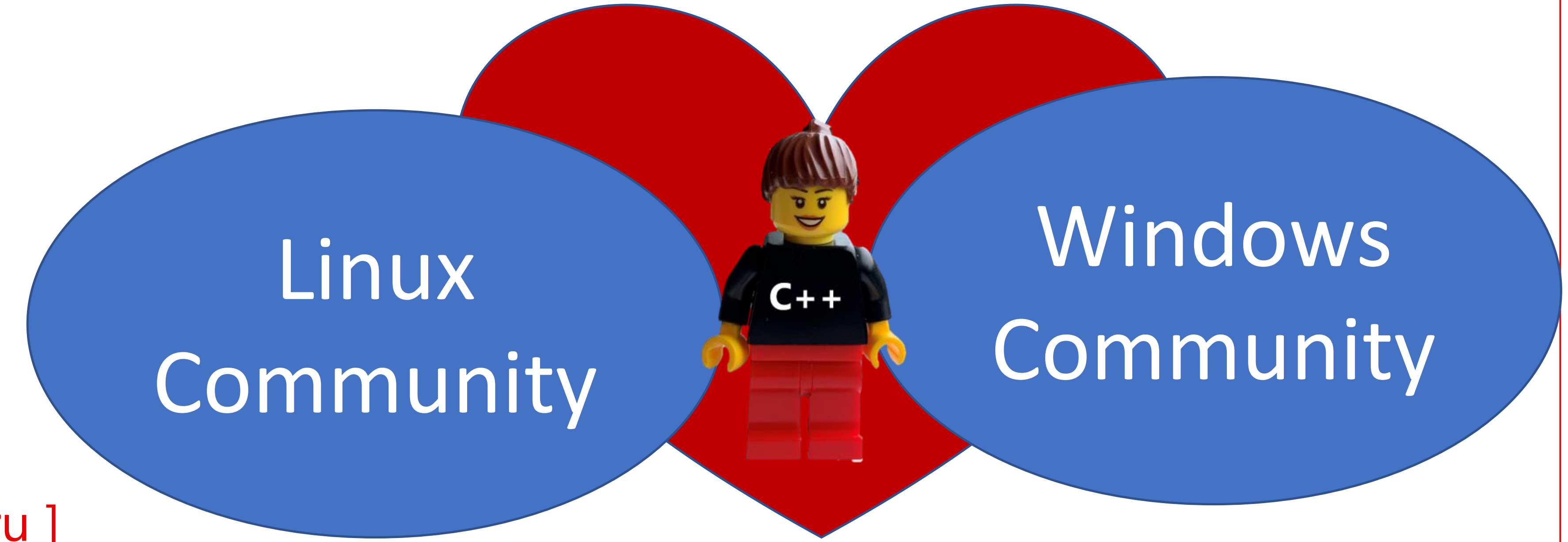
Visual Studio Code mit C++ Entwicklung und Package Management

Advanced Developers Conference ++
Development for Professionals!



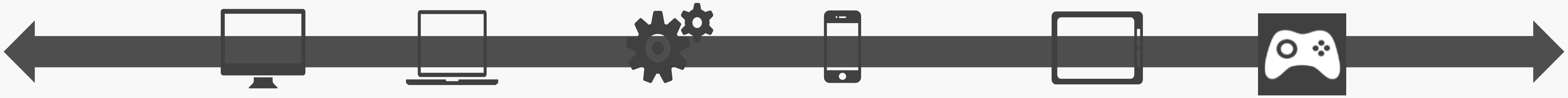
[Johannes Cosmin Dumitru]
[EMEA Chief Cloud Architect]
@dumian

ppedv

Agenda

- Introduction - Modernize Native Applications
- Building Native Apps with VS Code & Containers
- DevSecOps Pipelines

The evolution of enterprise apps



Internal Business Users, Enterprise B2B Partners, and Traditional channel enablement

Consumers, Mobile Workforce, and New online channel enablement

Systems of Record : LOB Apps

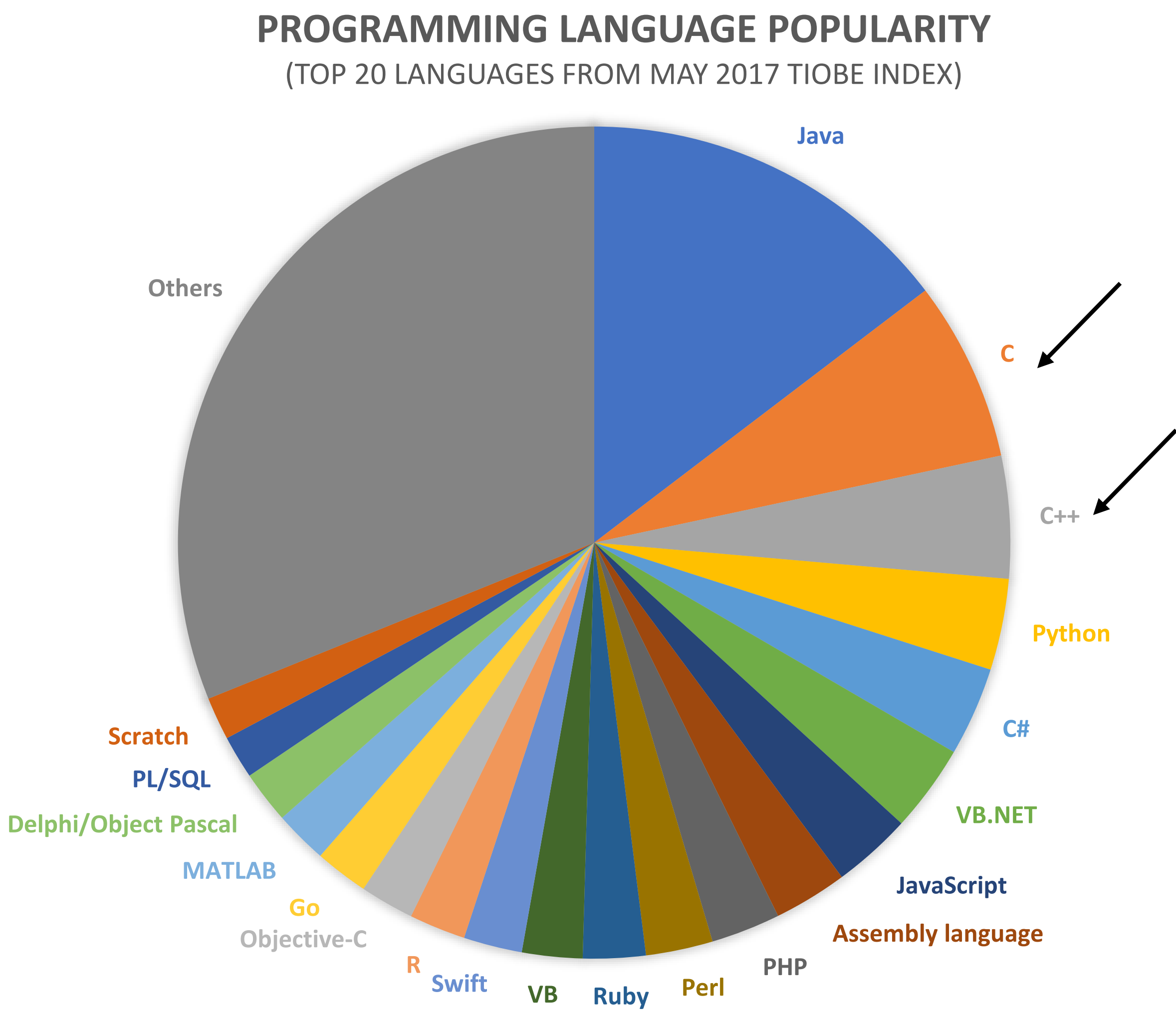
- Packaged solutions w/ customizations
- Longer release cycles
- Enterprise Scalability

Systems of Engagement : Modern Apps

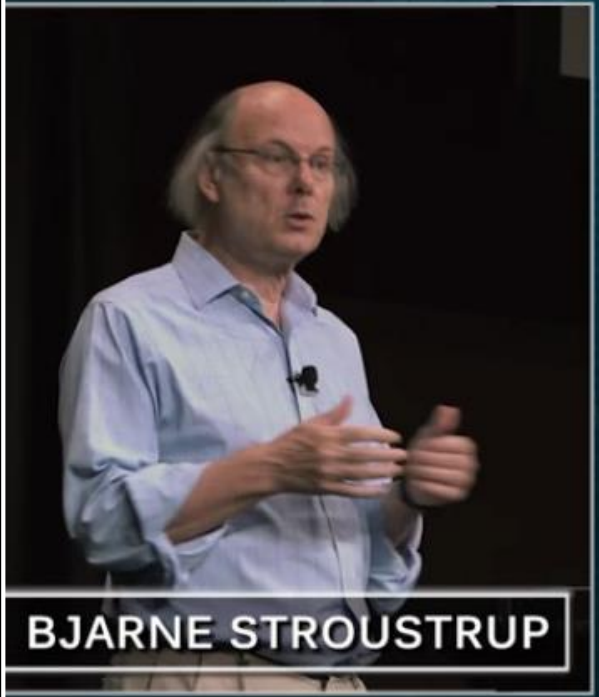
- Brand Differentiation
- Velocity : Continuous Delivery
- Enterprise Grade Internet Scalability

We live in a multilingual computing world

- Long tail of programming languages
- If you run more than a few apps you probably run more than a few languages
- We use different languages for different jobs:
  - R or Python for data science
  - Ruby or JavaScript for front ends
  - Java or C/C++ for server back ends
- Unfortunately, most runtimes can't run most languages very well



cppcon | 2017
THE C++ CONFERENCE • BELLEVUE, WASHINGTON



Learning
and
Teaching

Morgan Stanley

We need better package/build system

- How can a student install a GUI system and a database in the second week of a first programming course?
  - Different libraries provide different build support
  - Different systems have different build support
  - Different libraries don't interoperate well
- A dozen incompatible package managers is not a solution
  - Make simple tasks simple!

```
> download gui_xyz
> install gui_xyz
```

- Or equivalent (e.g., IDE)

Los geht's...

Packages

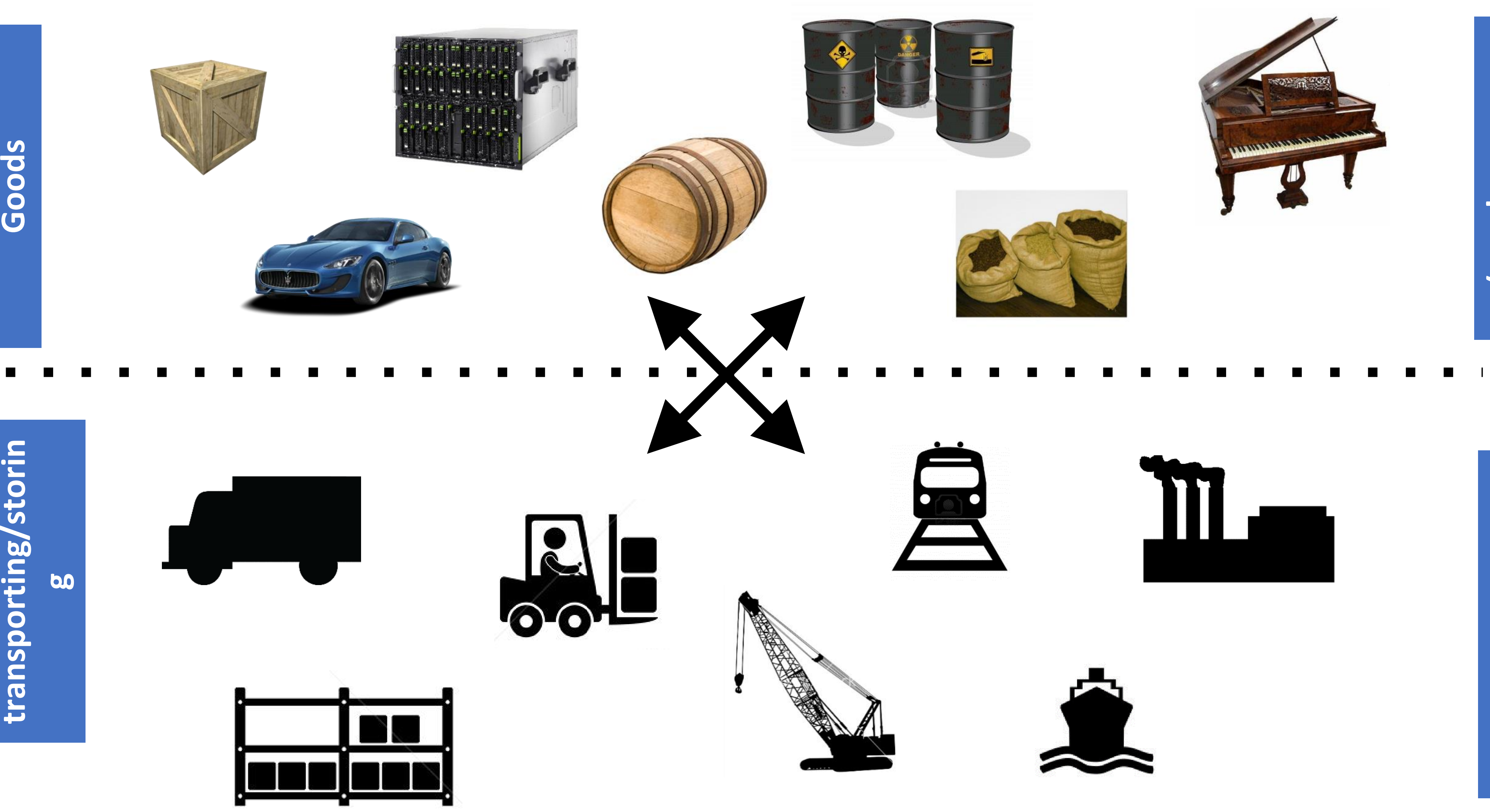
• Node	(npm)	600,000
• Python	(PyPI)	140,000
• Rust	(cargo)	18,000(16,000)
• C++	(?)	?

- CONAN
- HUNTER
- BUCKAROO
- VCPKG
- CGET
- CPM

An Inspiration: Cargo Transport Pre-1960

Multiplicity of Goods

Do I worry about how goods interact (e.g. coffee beans next to spices)



Multipilicity of methods for transporting/storin g

Can I transport quickly and smoothly (e.g. from boat to train to truck)

The Problem in 2014: Distributed Applications

Multiplicity of Stacks

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB

postgresql + pgv8 + v8

Queue

Redis + redis-sentinel

Analytics DB

hadoop + hive + thrift + OpenJDK

Background workers

Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend

Ruby + Rails + sass + Unicorn

API endpoint

Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

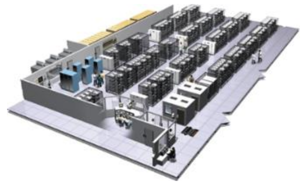
Multiplicity of hardware environments

Development VM

QA server

Public Cloud

Disaster recovery



Production Cluster

Customer Data Center

Production Servers

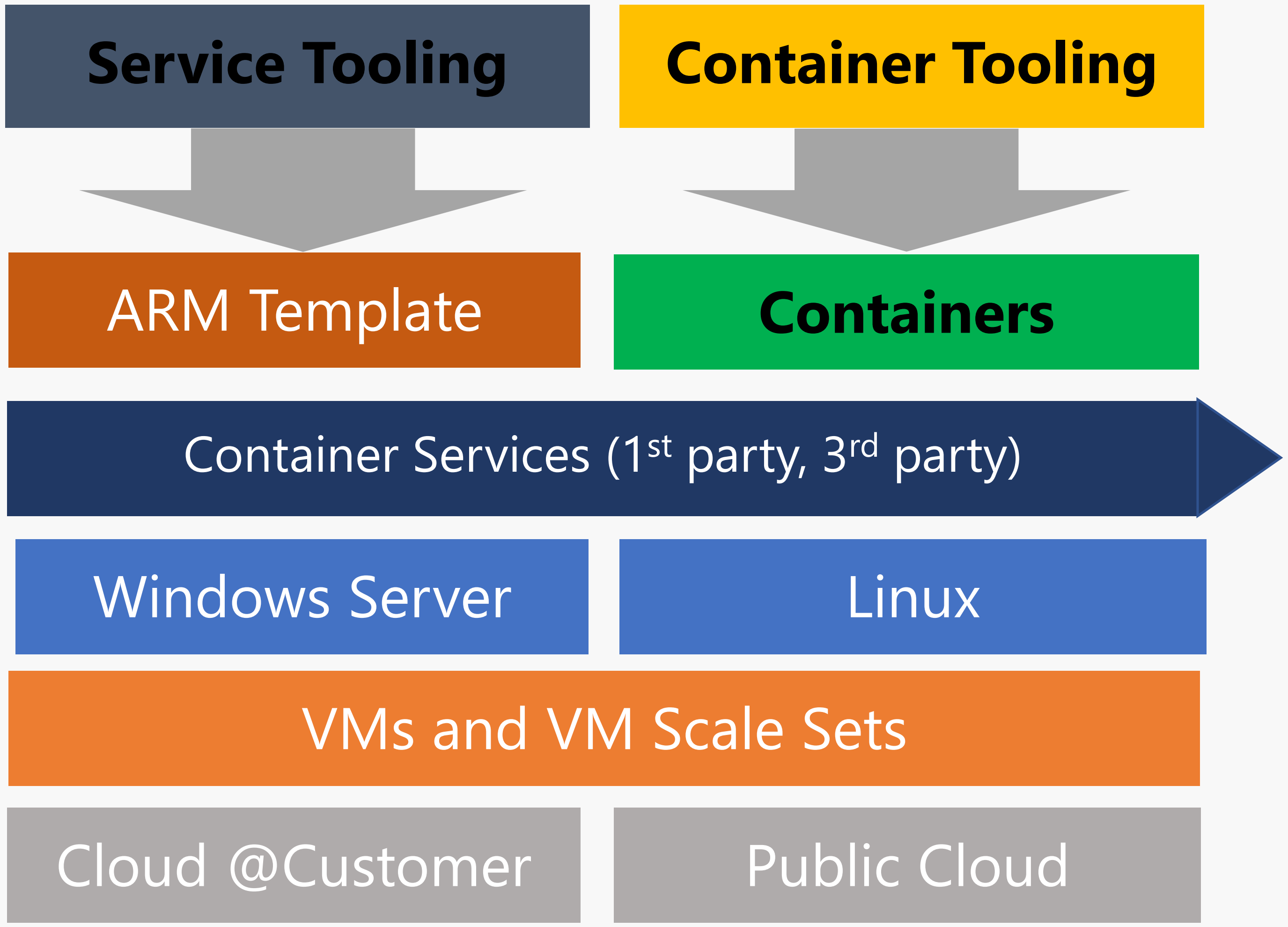
Contributor's laptop

Can I migrate smoothly and quickly?

The Intermodal Shipping Container Ecosystem

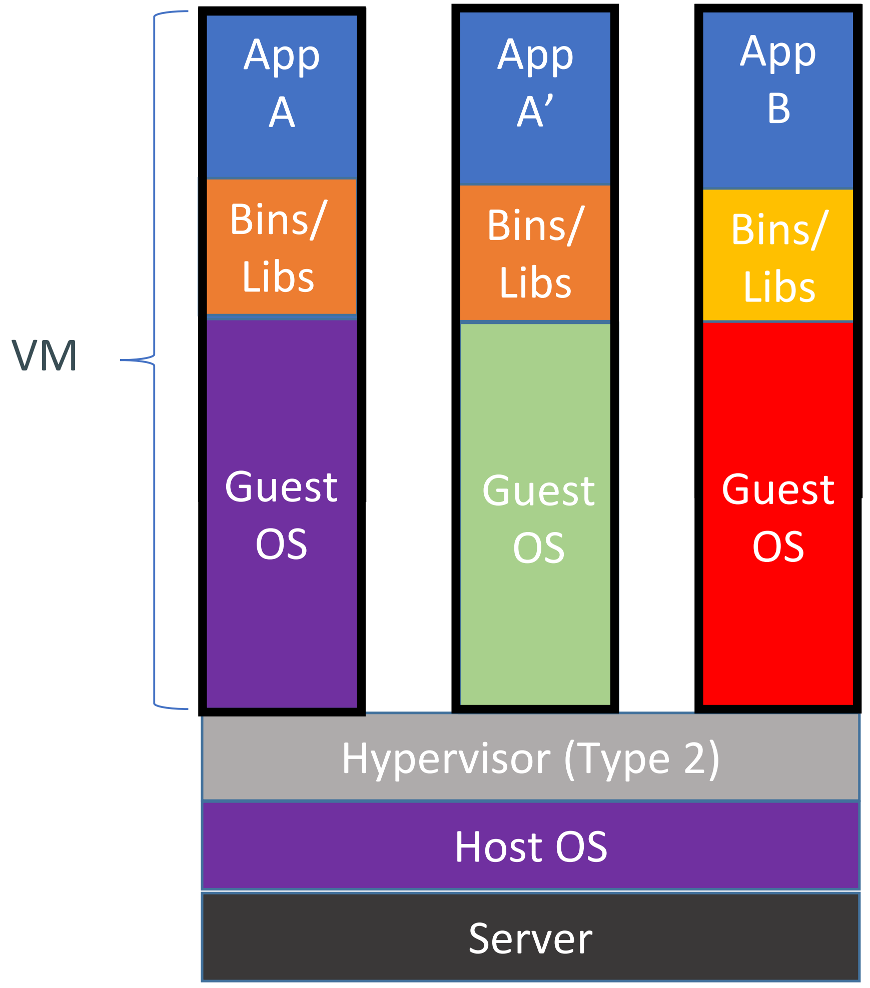
- 90% of all cargo now shipped in a standard container
- Order of magnitude reduction in cost and time to load and unload ships
- Massive reduction in losses due to theft or damage
- Huge reduction in freight cost as percent of final goods (from >25% to <3%)
- massive globalization
- 5000 ships deliver 200M containers per year

Layered for flexibility and agility



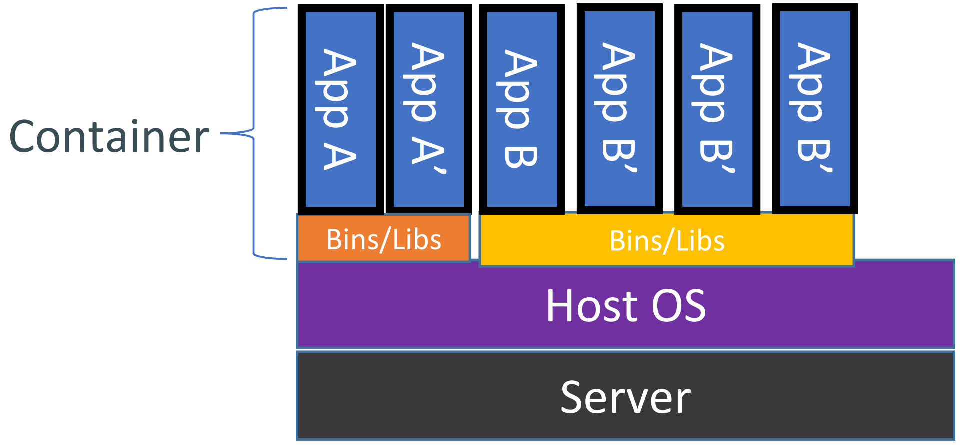
Layer	Supported Technologies
Configuration as Code	ARM, Dockerfile, Docker Compose, Marathon.json
Host cluster management	VM Scale Sets
Container orchestration	Docker Swarm, Chronos, Marathon, Apache Mesos
Monitoring	OMS, Statsd
Networking	IP per container
Storage	Persistent storage
???	

Comparison: Containers vs. VMs

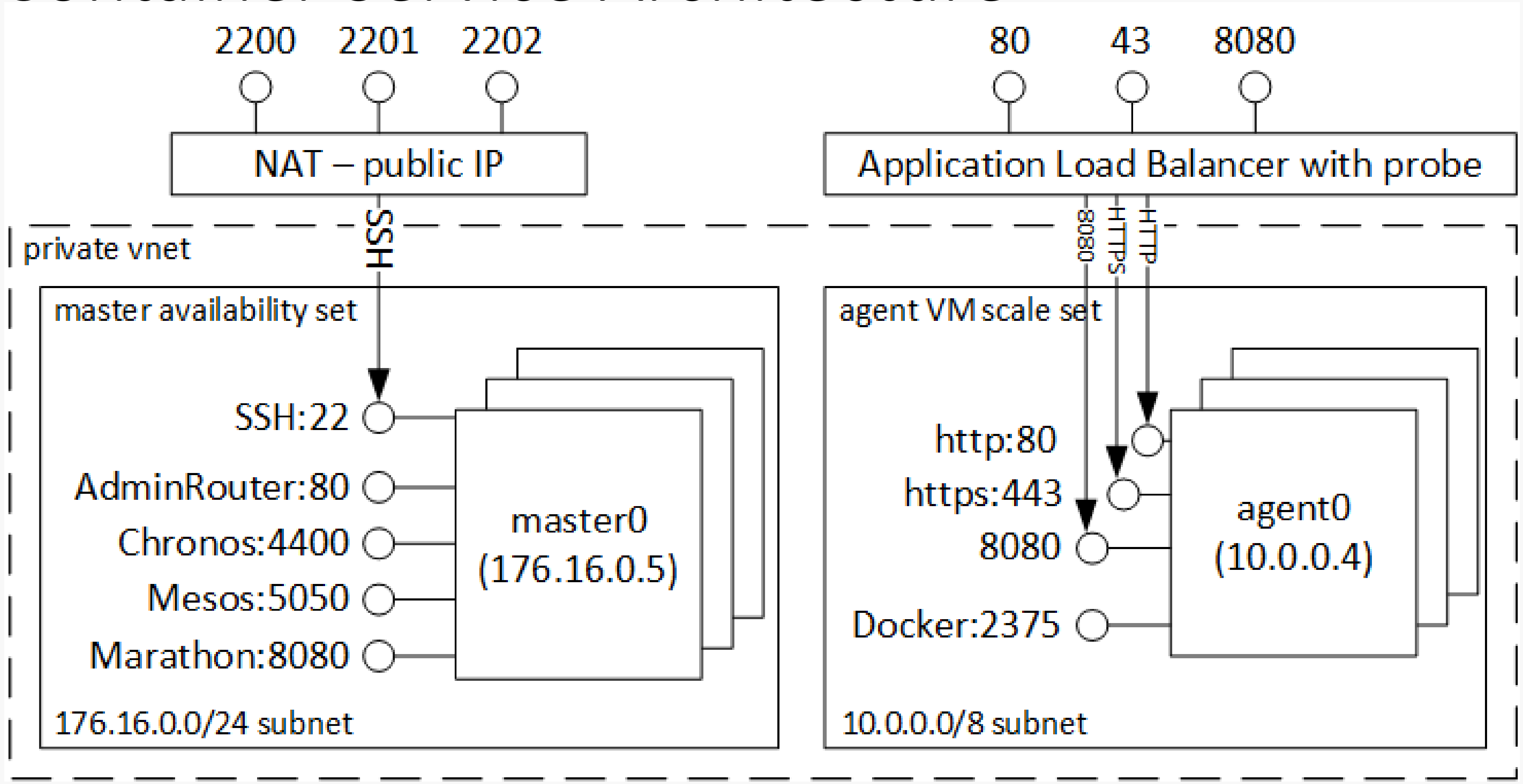


Containers are isolated, but share OS kernel and, where appropriate, bins/libraries

…result is significantly faster deployment, much less overhead, easier migration, faster restart



Container Service Architecture



DOCKER IMAGES

- Multiple layers on a union filesystem
- Images are immutable
- Each layer is cached for a given input
- Layers are built in series
- Multi-stage builds can build layers independently
- Files use copy-on-write when modified in subsequent layer
- Created using `docker build`

DOCKERFILE

- Documentative DSL to define steps to build an image
- Defines a base image layer
- Defines Layers that can copy files into the image being built or run shell commands in the image's context
- Specifies default env vars, ports, volumes, and a shell command to run at execution time

```
FROM debian:stretch-slim

RUN dpkg --add-architecture i386

RUN apt-get update && apt-get install -y --no-install-recommends \
        gcc \
        libc6-dev \
        make \
        \
        libc6-dev:i386 \
        libgcc-6-dev:i386 \
        \
        libc6-dev-arm64-cross \
        libc6-dev-armel-cross \
        libc6-dev-armhf-cross \
        libc6-dev-ppc64el-cross \
        libc6-dev-s390x-cross \
        \
        gcc-aarch64-linux-gnu \
        gcc-arm-linux-gnueabi \
        gcc-arm-linux-gnueabihf \
        gcc-powerpc64le-linux-gnu \
        gcc-s390x-linux-gnu \
        \
        file \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /usr/src/hello
COPY . .

RUN set -ex; \
    make clean all test \
        TARGET_ARCH='amd64' \
        CC='x86_64-linux-gnu-gcc'\
        STRIP='x86_64-linux-gnu-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='arm32v5' \
        CC='arm-linux-gnueabi-gcc'\
        STRIP='arm-linux-gnueabi-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='arm32v7' \
        CC='arm-linux-gnueabihf-gcc'\
        STRIP='arm-linux-gnueabihf-strip'

RUN set -ex; \
    make clean all \
```

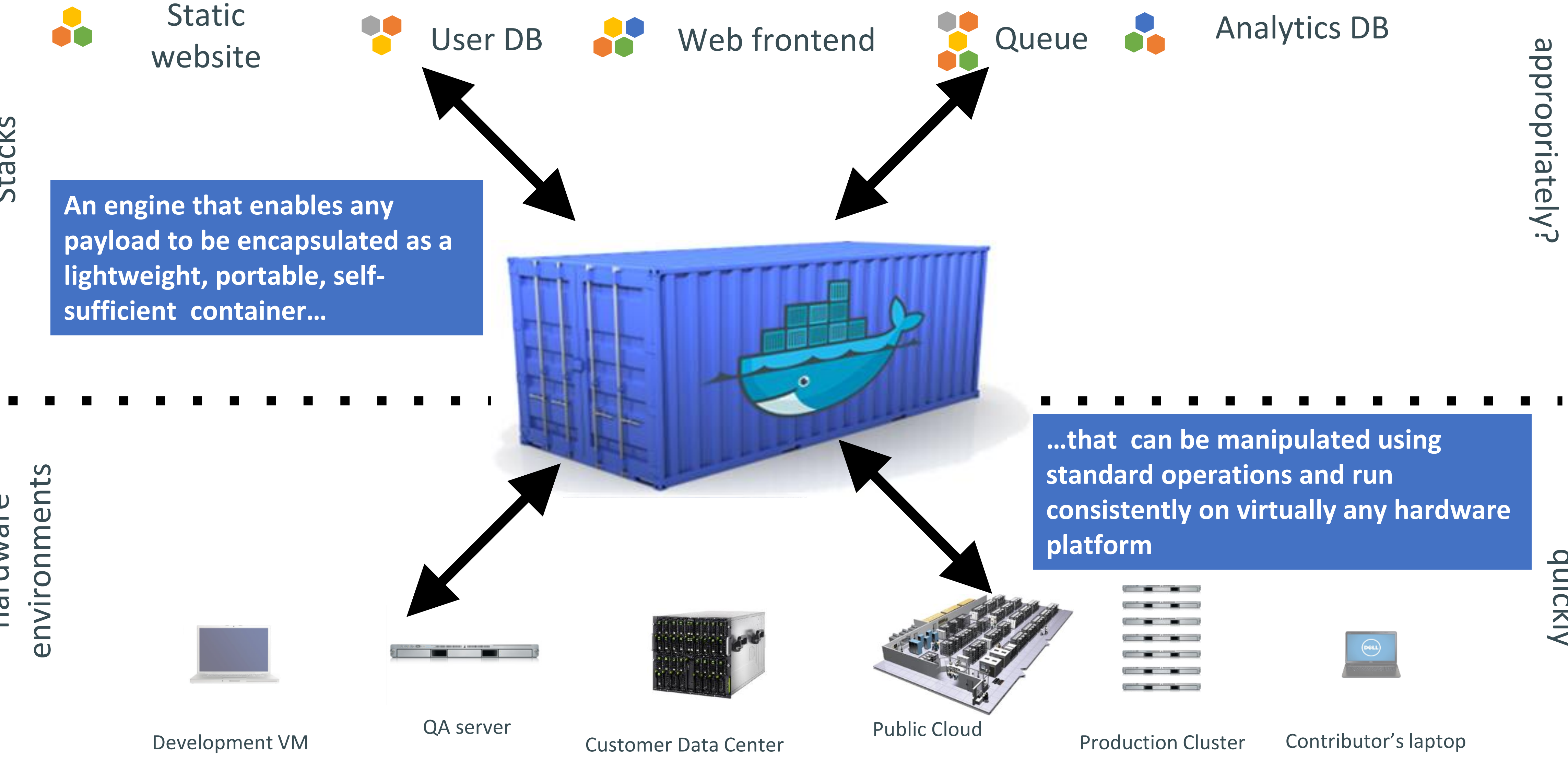
HELLO WORLD C++ STYLE - DOCKERFILE

```
FROM dumians/cppdock:adccpp20

COPY main.cpp /src/build/

WORKDIR /src/build

RUN g++ -stdlib=libc++ -lc++abi main.cpp

CMD ["./a.out"]
```

Let’s create an **ecosystem** for **distributed** applications

Multiplicity of Stacks

Do services and apps interact appropriately?



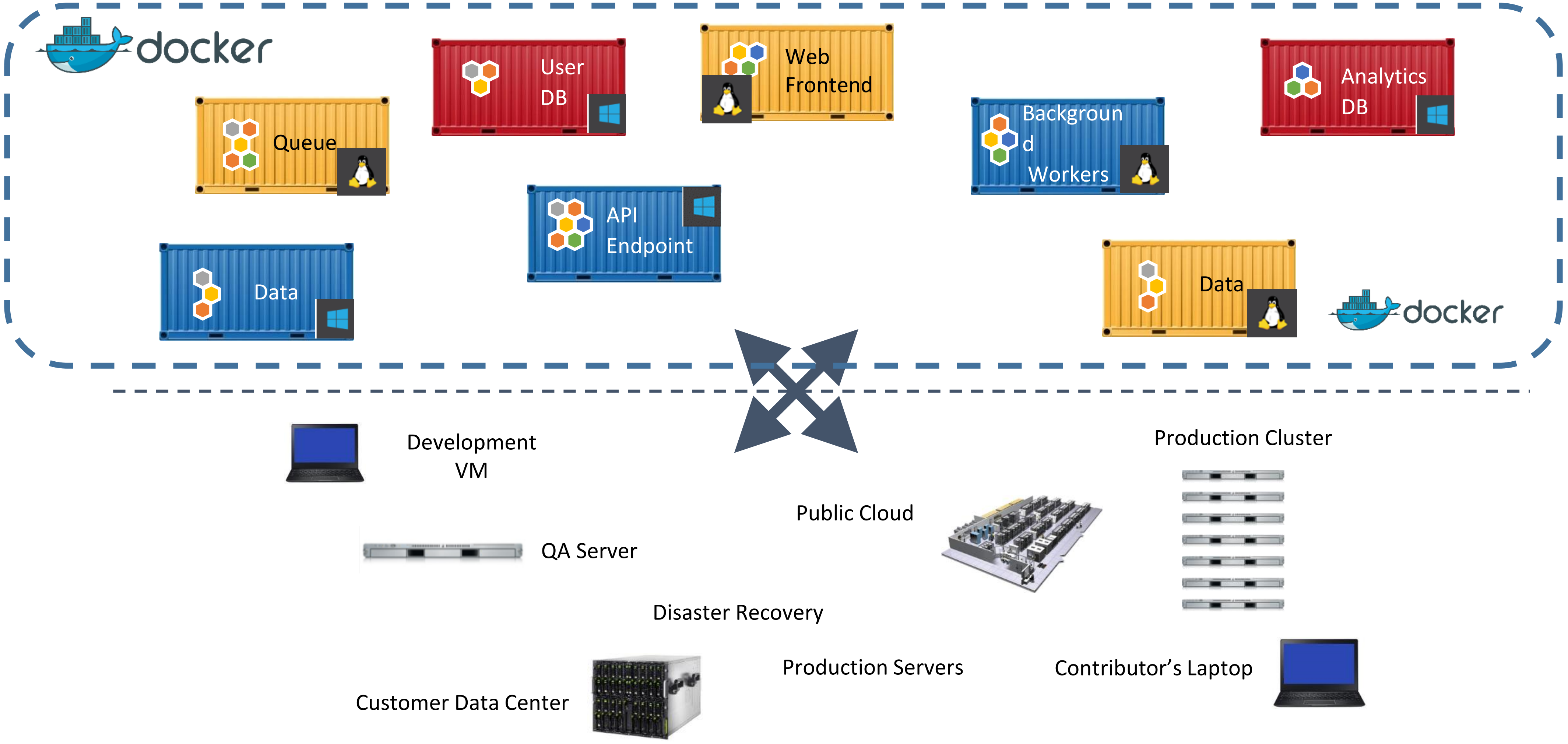
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

Multiplicity of hardware environments

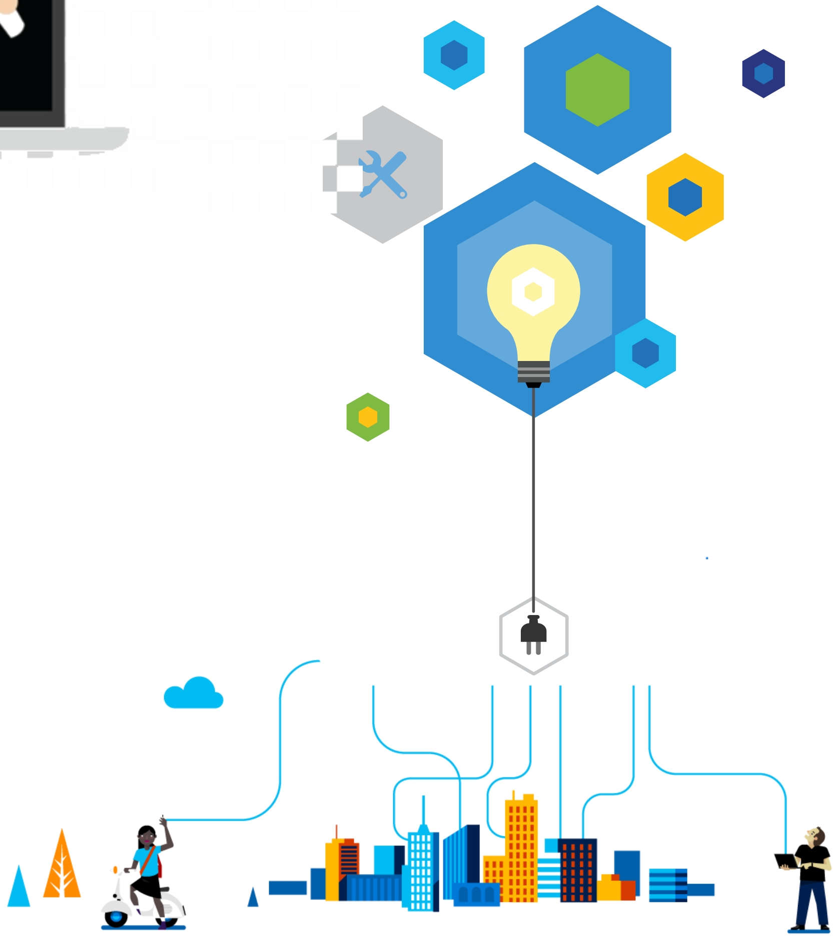
Can I migrate smoothly and quickly

Distributed Applications With Both Linux and Windows Components



Demo

- VS Code
  - C++ Development
  - Container management



Visual Studio Code: C/C++ Extension

- Lightweight, keyboard focused
- Git integration
- Code Editing
  - IntelliSense, Code Browsing, Switch header/source, Code formatting (clang-format)
- Debugging
  - Core-dump debugging, launch, attach, breakpoints (incl. conditional and function), stepping, threads, call stack, watch, GDB and MI commands
- Easily run, build, test, and run external tasks

https://code.visualstudio.com/docs/languages/cpp

Vcpkg, Conan,CGET, CPM : An open source tools

80% of C++ projects use 3+ 3rd party libs

A majority of them use open source libraries

Simplifying rebuilding libs on Windows

A simple cmd line: Usage: **vcpkg install boost**

Installs the .h, .lib and binaries in a “lib folder” ready to use and to deploy

Open source tool based on a port tree approach (Vcpkg, Conan)

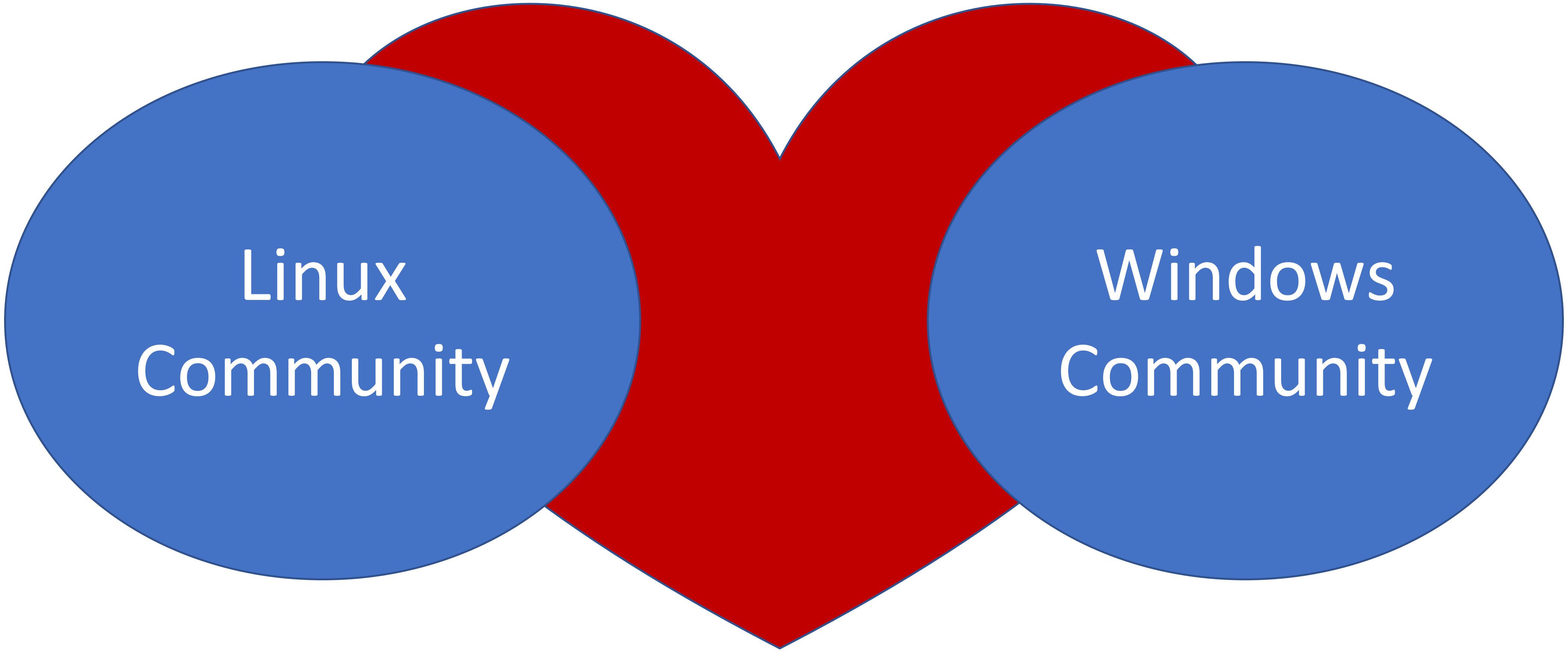
Port file tree is on GitHub, you can contribute to it and/or fork it

Conformance Testing with ~60 OSS Libraries from GitHub

- Testing with GitHub master branches and compiler development trunk
  - MSVC default mode – 58 projects
  - MSVC /std:c++17 mode – 58 projects
  - MSVC /permissive- mode – 55 projects

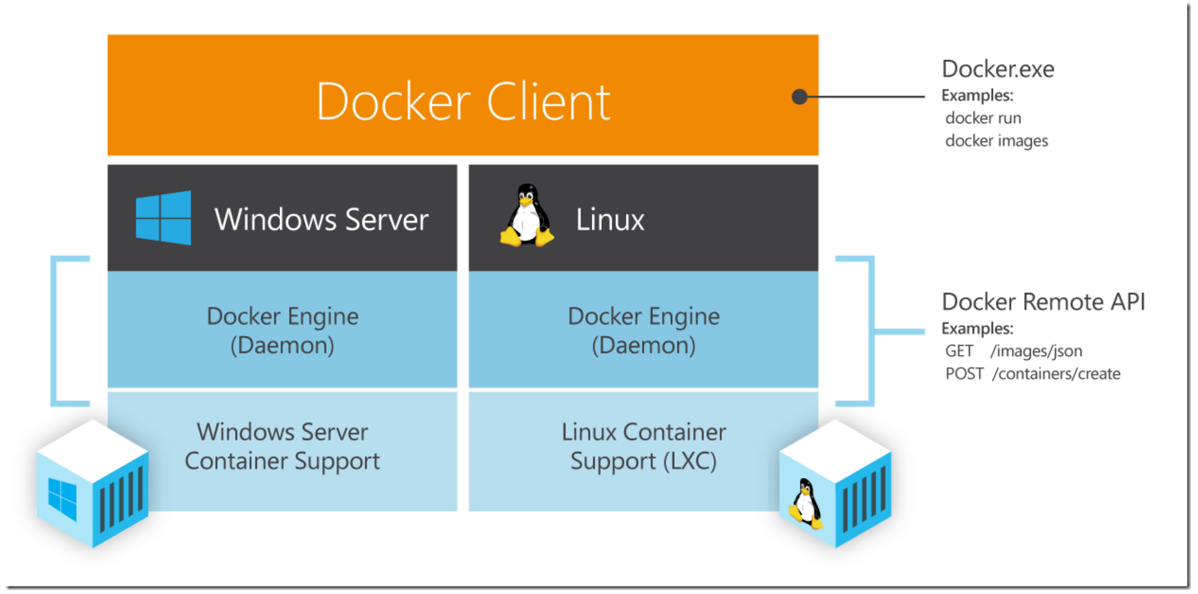
No.	Source
1	CoreCLR
2	Chakra
3	ClangLLVM
4	OpenSSL
5	Chrome
6	OpenCV
7	RxCpp
8	Boost
9	UnrealEngine
10	Electron
11	QTCreator
12	QT
13	Cocos2dx
14	OSQuery
15	FLAC
16	WinRT
17	Z3
18	PDFium
19	X265
20	RocksDB
21	VCPKG
22	PostgreSQL
23	CryEngine
24	APPLE_LZFSE
25	Blender
26	Dolphin
27	Facebook_ZSTD
28	Glslang
29	Google_Brotli
30	Google_LiquidFun
31	Google_MathFu
32	Google_Protobuf
33	Google_RE2
34	Google_Snappy
35	Google_VP9
36	Google_SwiftShader
37	Irrlicht
38	LAME
39	ITK
40	VTK
41	Sprout
42	LibGIT2
43	LibJPEG
44	LibJPEG_Turbo
45	LUA
46	LUAJIT
47	LZ4
48	Serious_Engine
49	Python3
50	PHP7
51	MySQL
52	Mesos
53	SDL
54	Azure_iot_sdk_c
55	Dlib
56	Bond
57	KTL
58	Outcome

Bringing Communities Together

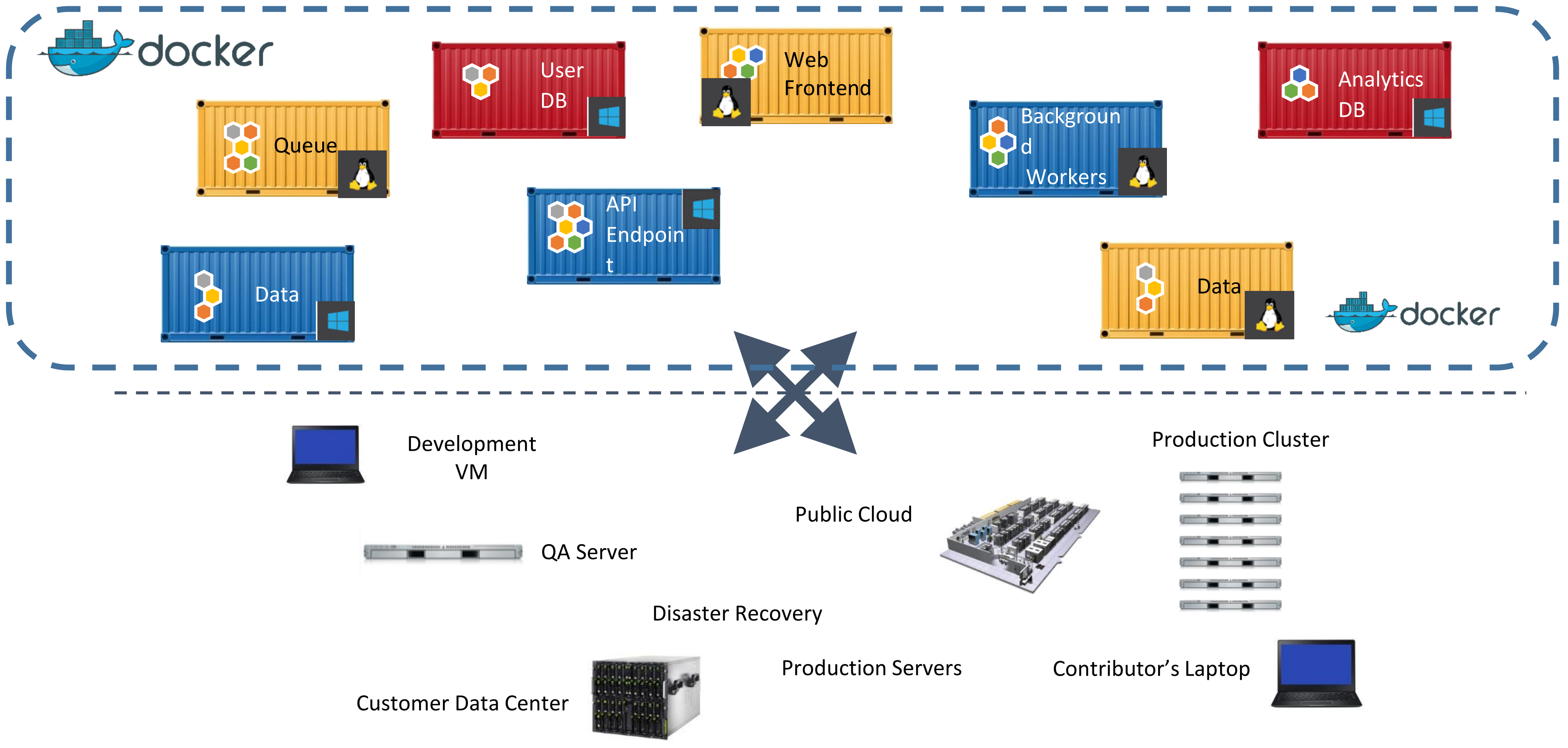


Recent: Docker for Windows

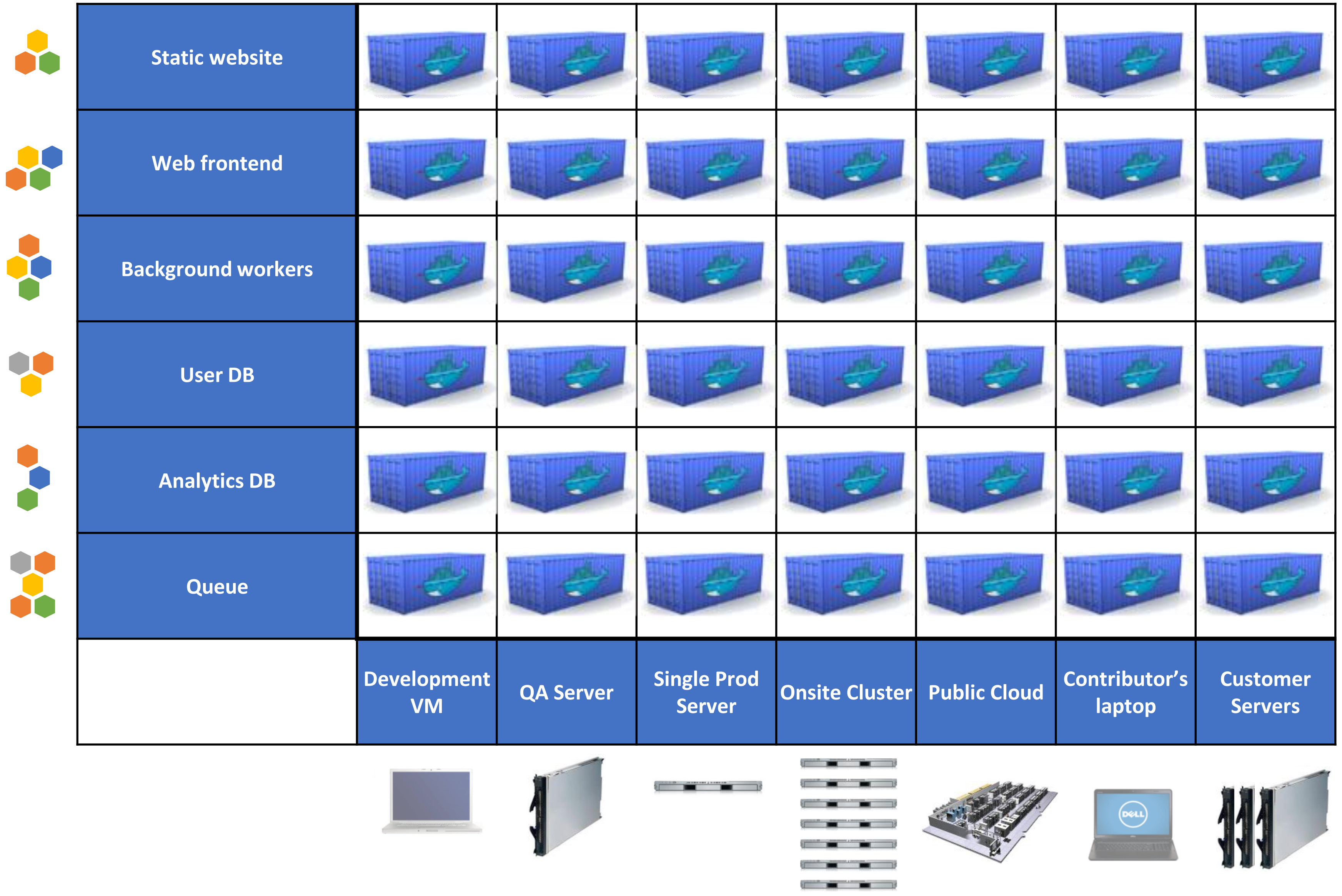
- Bring Docker and Containers to Windows
- Contribute to open source Docker Engine to support Windows
- Local box support on Hyper-V



Distributed Applications With Both Linux and Windows Components

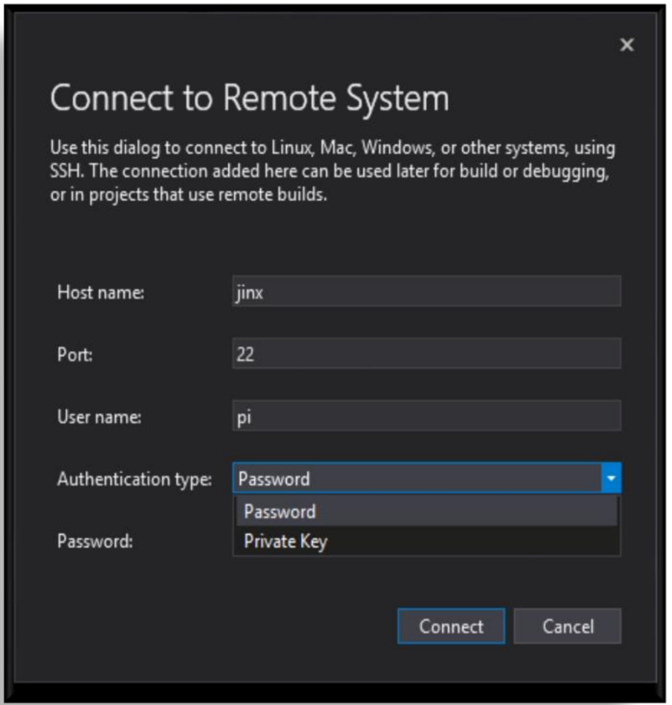


And eliminate the matrix from Hell



Linux

- Use Visual Studio with any Linux distro or Windows Subsystem for Linux (WSL)
  - Remote system needs SSH, GDB, and GCC for compile
  - Connect using user/password or private key
  - Project templates enable control of GCC/GDB on remote target
  - IntelliSense supports GCC with standard Linux libraries out of the box
  - Debug from your projects or attach to remote process
    - Use either gdb or gdbserver on the remote
    - Python pretty printer type visualizers supported in gdb mode
  - Support for CMake > 3.8 added in 15.4
- Resources
  - Documentation: https://aka.ms/vslinux
  - Issues, discussion: https://github.com/microsoft/vslinux



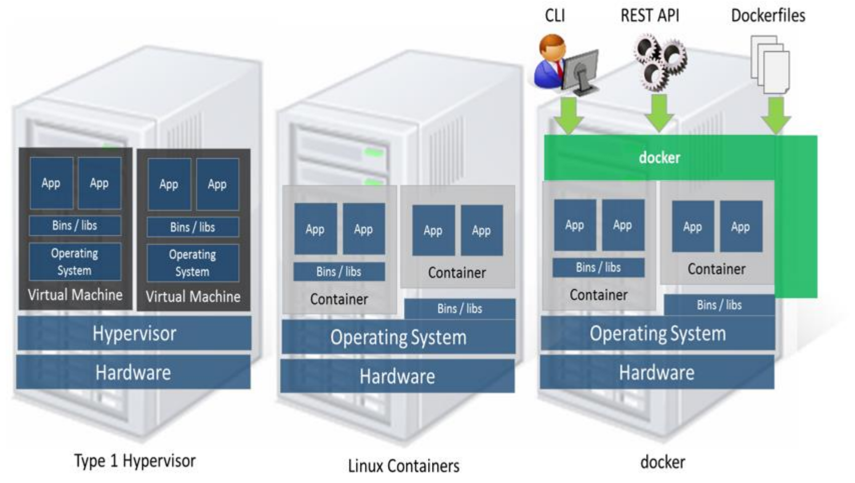
Connect to Remote System

Use this dialog to connect to Linux, Mac, Windows, or other systems, using SSH. The connection added here can be used later for build or debugging, or in projects that use remote builds.

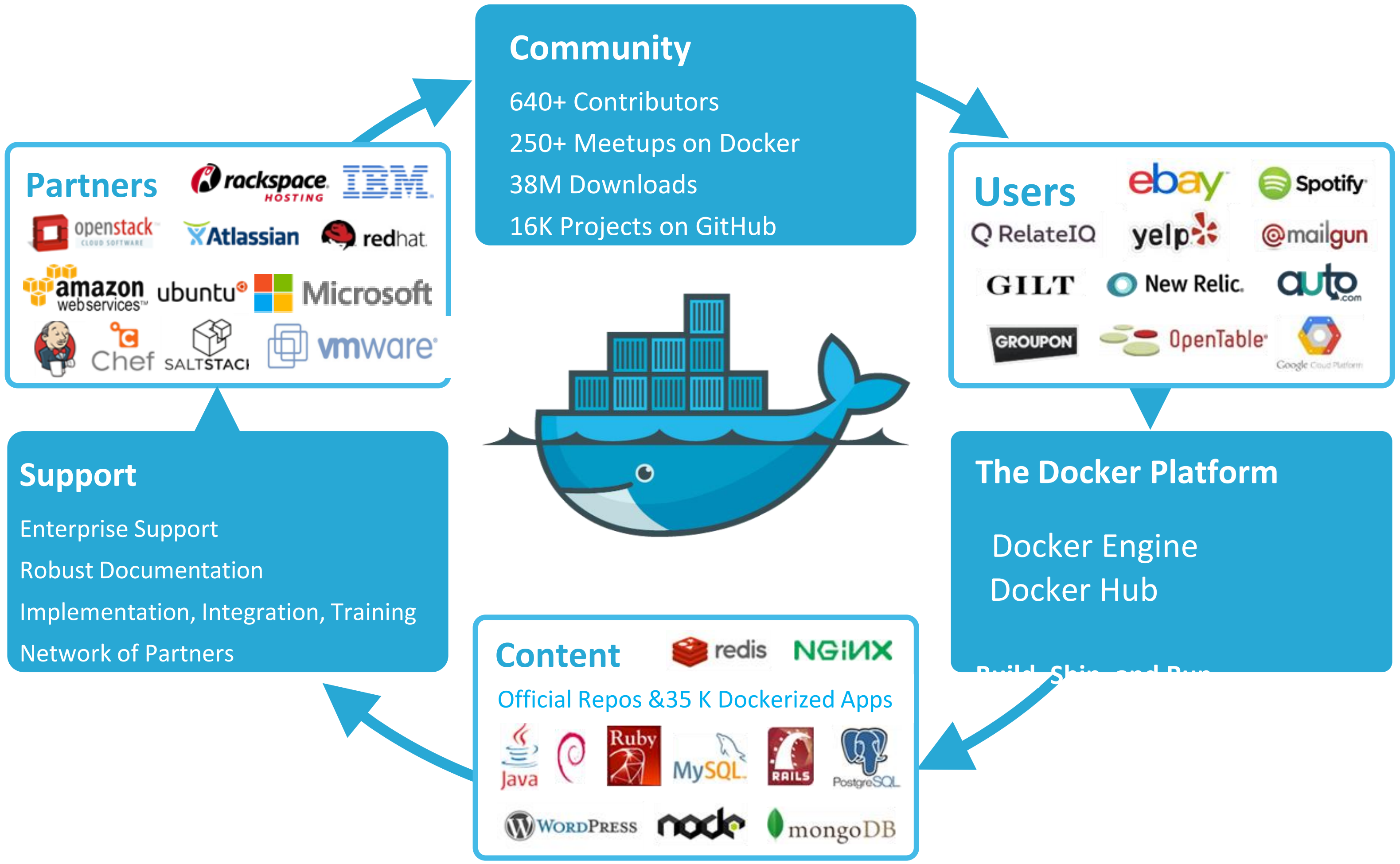
Host name:	jinx
Port:	22
User name:	pi
Authentication type:	Password
Password:	

Importance of an Ecosystem

- Container technology has been around for a while (LXC, Solaris Zones, BSD Jails)
- Analogy: Shipping containers are not just steel boxes
- With Docker, low level containers get the following:
  - Re-usable components
  - Ability to run on any Linux server today: physical, virtual, VM, cloud, OpenStack, +++
  - Ability to move between any of the above in a matter of seconds-no modification or delay
  - Ability to share containerized components
  - Self contained environment—no dependency hell
  - Tools for how containers work together: linking, nesting, discovery, orchestration
- “Containerization” is really “Dockerization”



Snapshot: The Docker Ecosystem



Community

640+ Contributors
250+ Meetups on Docker
38M Downloads
16K Projects on GitHub

Users

The Docker Platform

Docker Engine
Docker Hub

Build, Ship, and Run

Content

Official Repos &35 K Dockerized Apps

Support

Enterprise Support
Robust Documentation
Implementation, Integration, Training
Network of Partners

Partners

Images and Containers

- Dockerfile
- Hello World
- Multi-Stage Build
- Build a Toolchain
- Contributing to an Open Source Project
- CppDock

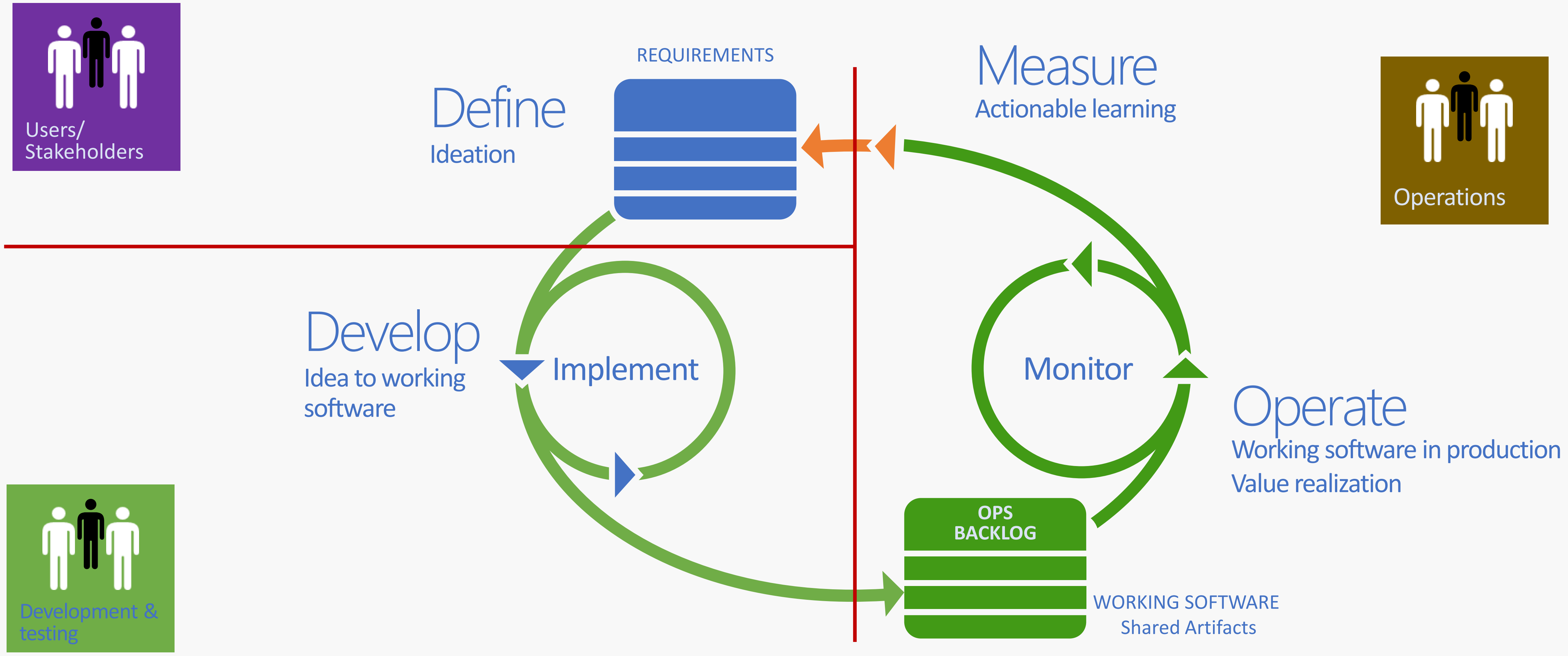
Advanced Developers Conference C++
Development for Professionals!

Demo

ppedv

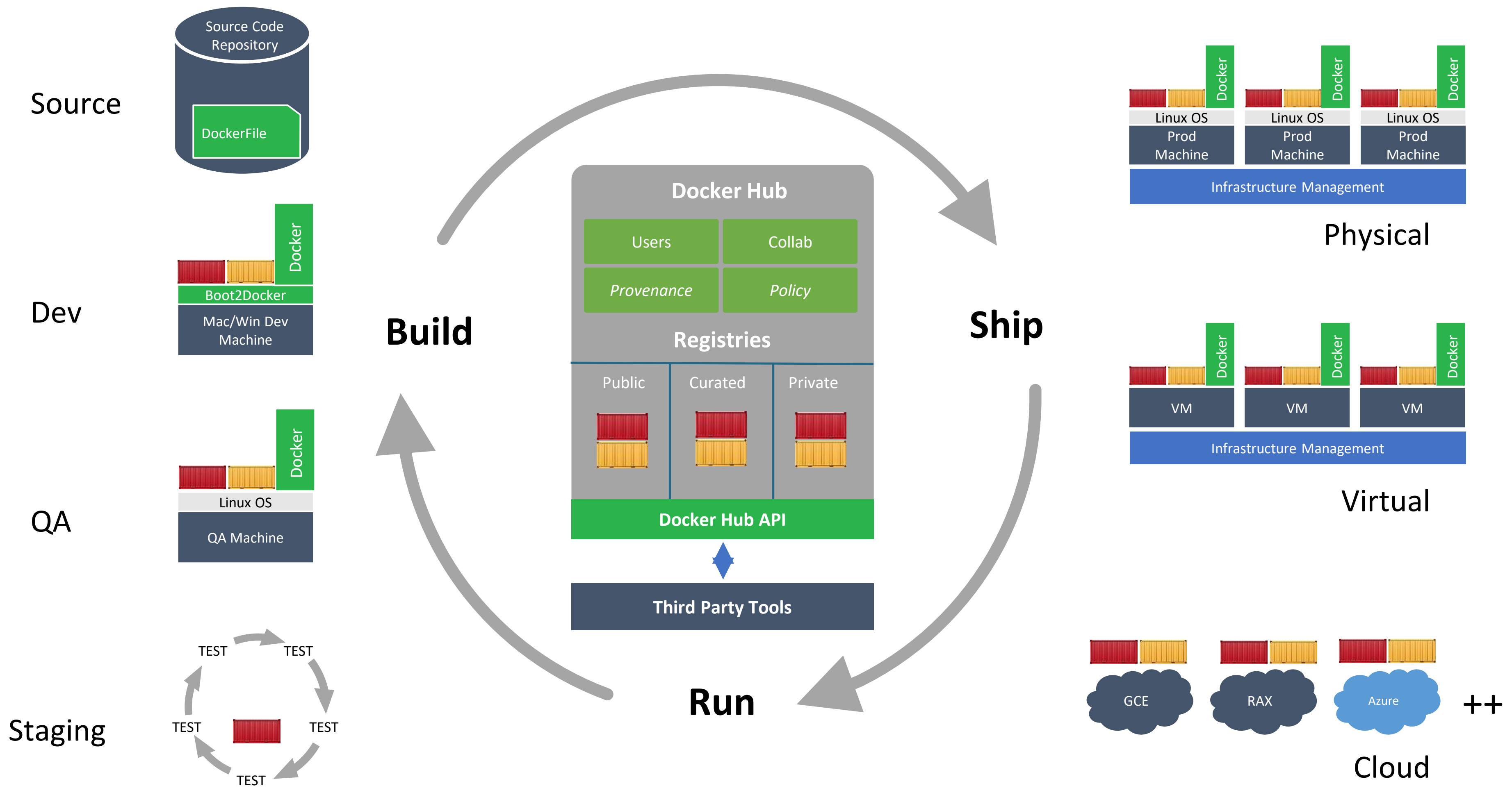
Modern Apps Life Cycle

Waste elimination | Cycle time reduction | Integration & Visibility



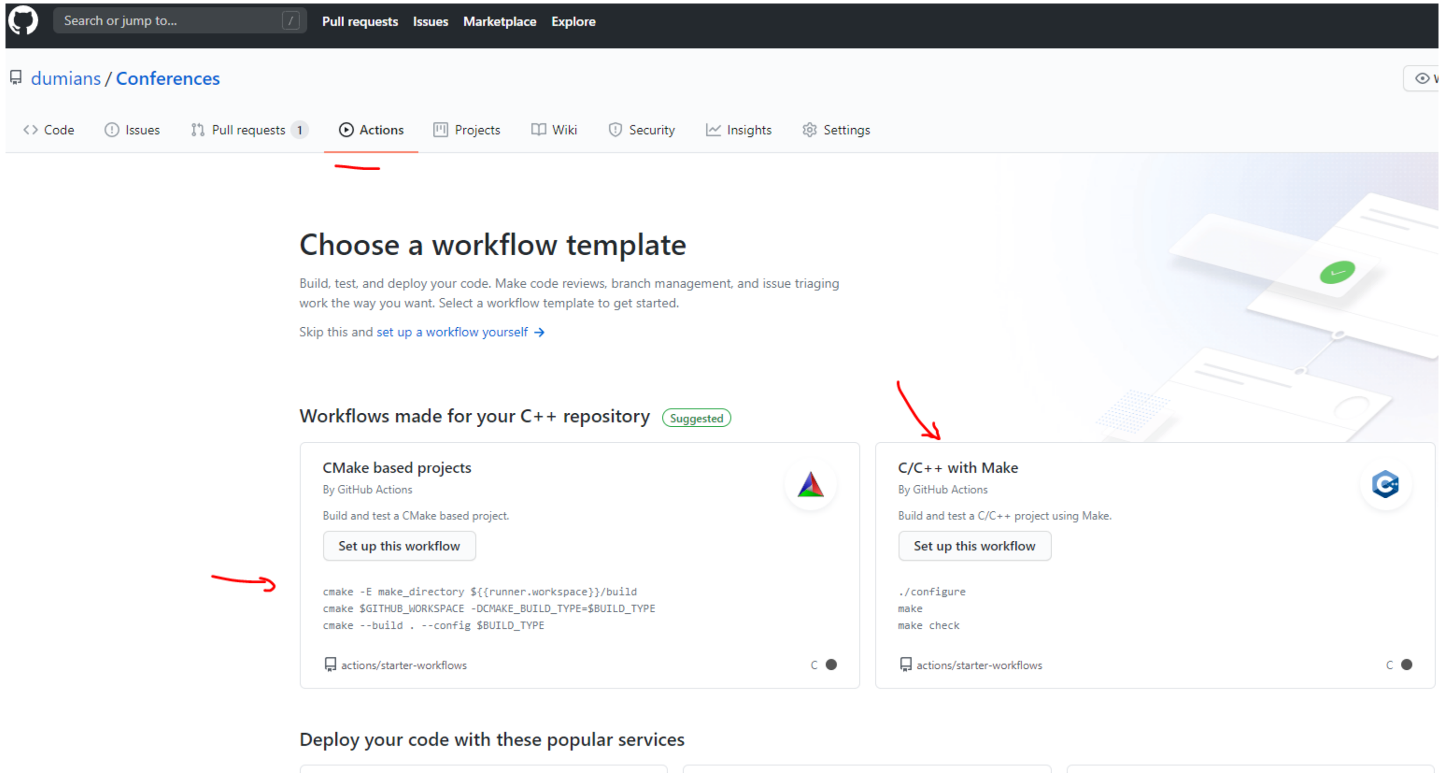
Continuous feedback | Continuous quality | Continuous delivery

Container Registry : Build, Ship, Run Applications



Docker Hub, Oracle Cloud Registry, Azure Container Registry

DevOps Pipelines



Choose a workflow template

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow template to get started.

Skip this and set up a workflow yourself →

Workflows made for your C++ repository (Suggested)

CMake based projects
By GitHub Actions
Build and test a CMake based project.
Set up this workflow

```
cmake -E make_directory ${{runner.workspace}}/build
cmake $GITHUB_WORKSPACE -DCMAKE_BUILD_TYPE=$BUILD_TYPE
cmake --build . --config $BUILD_TYPE
```

actions/starter-workflows C

C/C++ with Make
By GitHub Actions
Build and test a C/C++ project using Make.
Set up this workflow

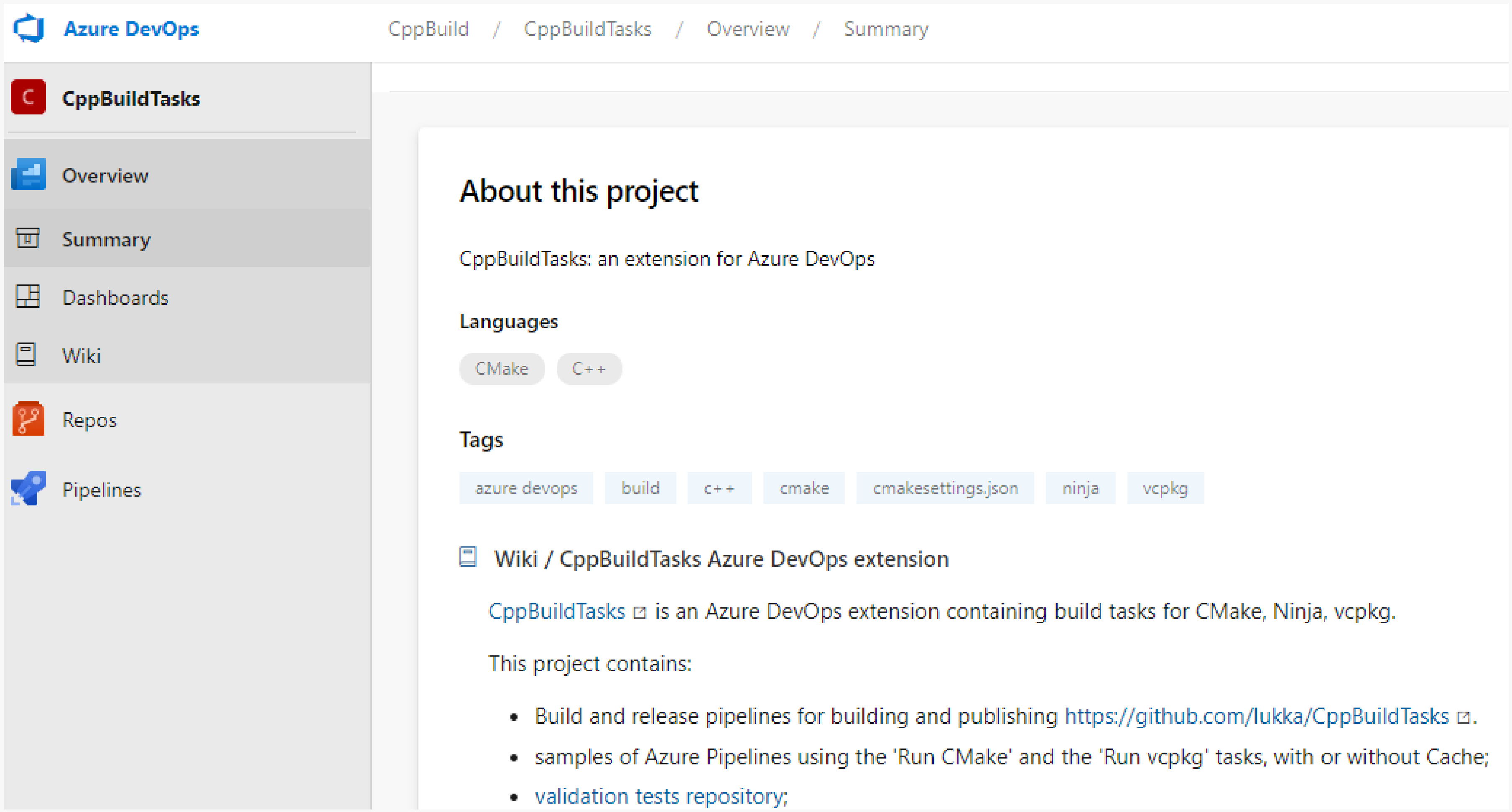
```
./configure
make
make check
```

actions/starter-workflows C

Deploy your code with these popular services

- Github Actions
- GitLab
- Azure DevOps

Pipelines extensions



About this project

CppBuildTasks: an extension for Azure DevOps

Languages

CMake C++

Tags

azure devops build c++ cmake cmakesettings.json ninja vcpkg

Wiki / CppBuildTasks Azure DevOps extension

CppBuildTasks is an Azure DevOps extension containing build tasks for CMake, Ninja, vcpkg.

This project contains:

- Build and release pipelines for building and publishing https://github.com/lukka/CppBuildTasks.
- samples of Azure Pipelines using the 'Run CMake' and the 'Run vcpkg' tasks, with or without Cache;
- validation tests repository;

Summary

- Integrated build toolchain for C++
- Covers creation, development, testing, and delivery
- Uniform interface across platforms/compilers
- Archive and version control-based repositories
- Dependency-free, all you need is a C++ compiler

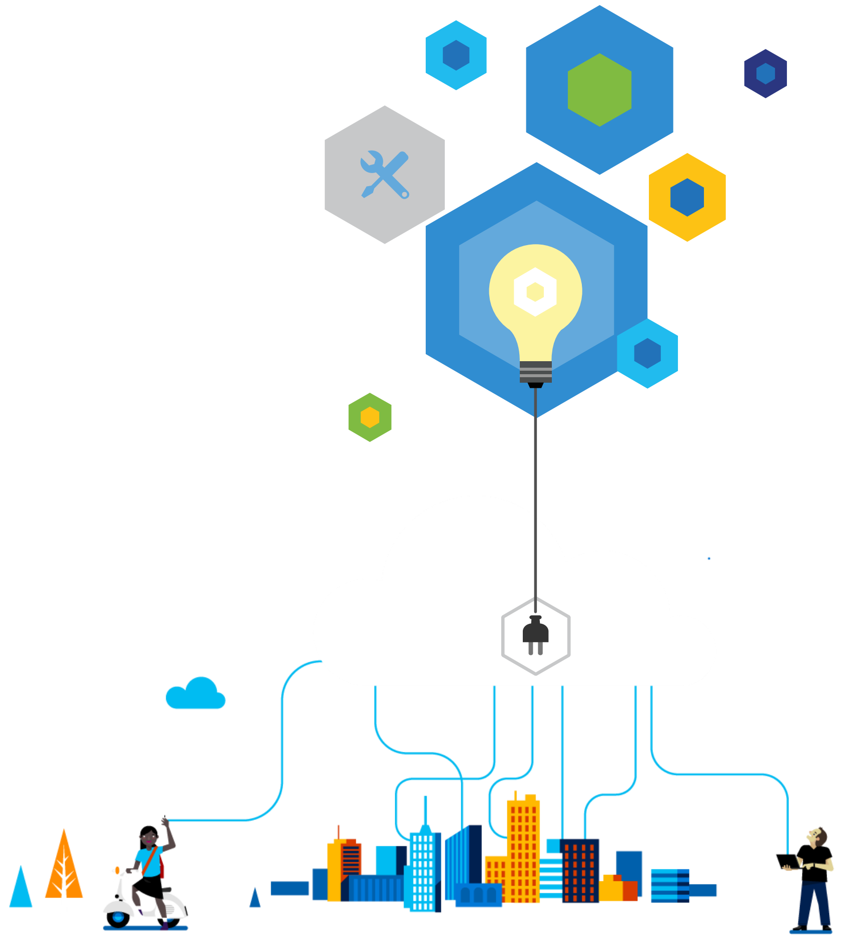
- genuuid
- genuuid--> libstud-uuid
- libstud-uuid --> cppget.org

linux-gcc_7.3	**macos_10.12-clang_9.1**
linux-gcc_8.2	macos_10.13-clang_10.0
linux-clang_5.0	macos_10.13-homebrew_gcc_8.1

How to get Started - Some References

- VCPKG
- Conan

https://code.visualstudio.com/docs/languages/cpp



Advanced Developers Conference C++
Development for Professionals!

Fragen?

ppedv

Vielen Dank!

Ich freue mich auf Feedback!