Lecture 4: 构建系统与现代软件栈

Build Systems & Modern Software Stack

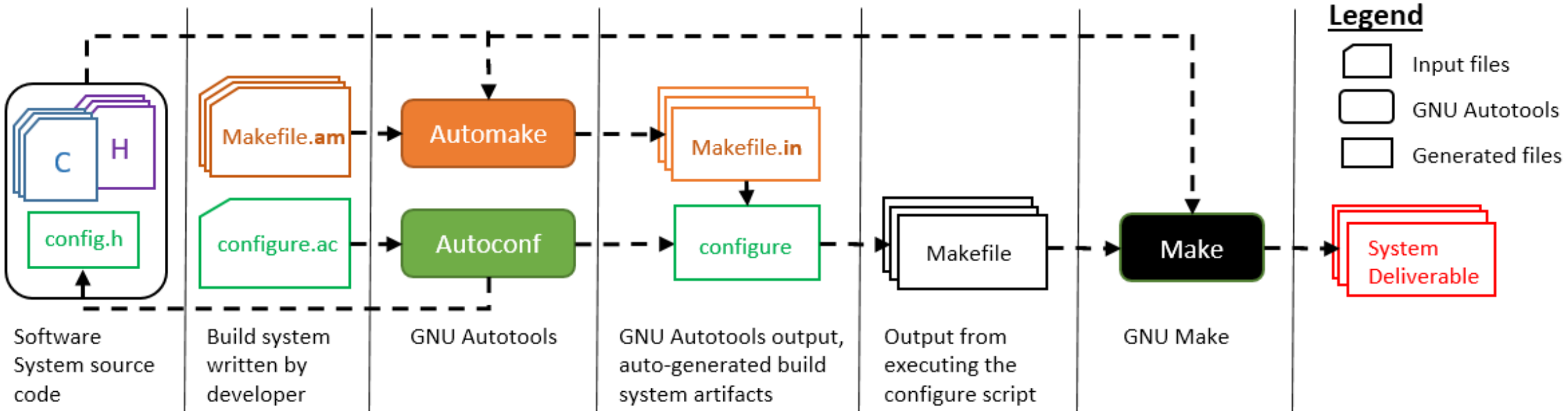
构建系统

自动化构建
持续集成
依赖

自动化构建

- 针对于源代码、环境、任务的一段特定流程
- 你需要定义的
  - 构建的目标
  - 构建所需的依赖
  - 构建时的规则
- 现代的构建工具
  - For C++/C: GNU Make, Ninja...
  - For Java: Maven, Gradle...
  - Build system generator: GNU Autotools, CMake...

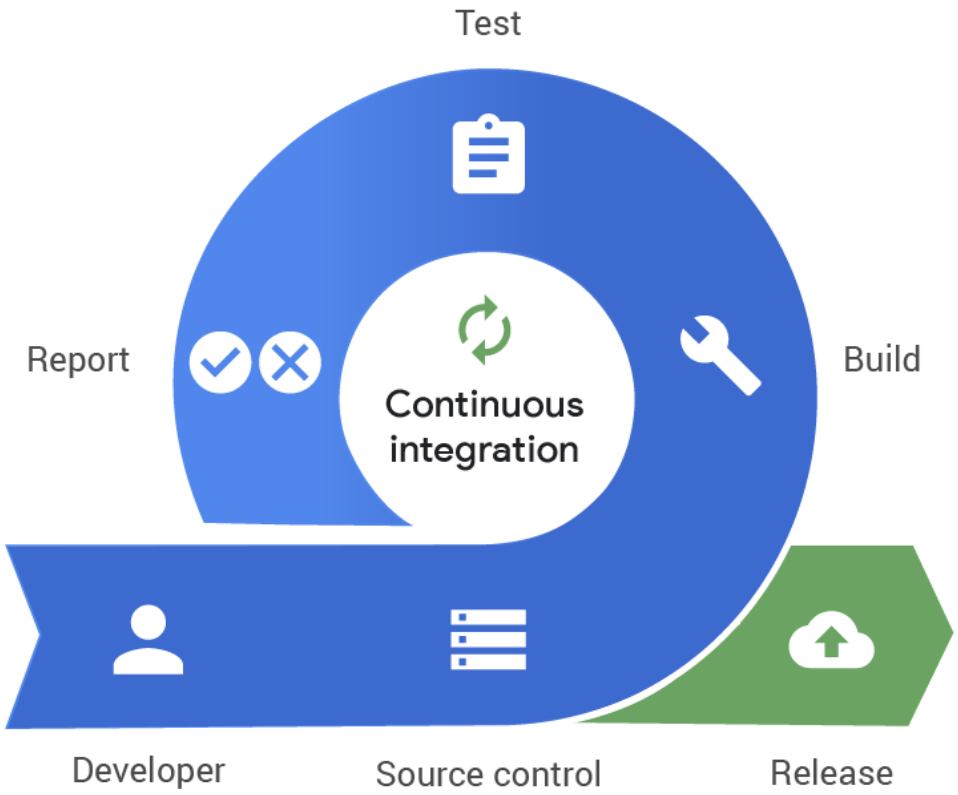
自动化构建

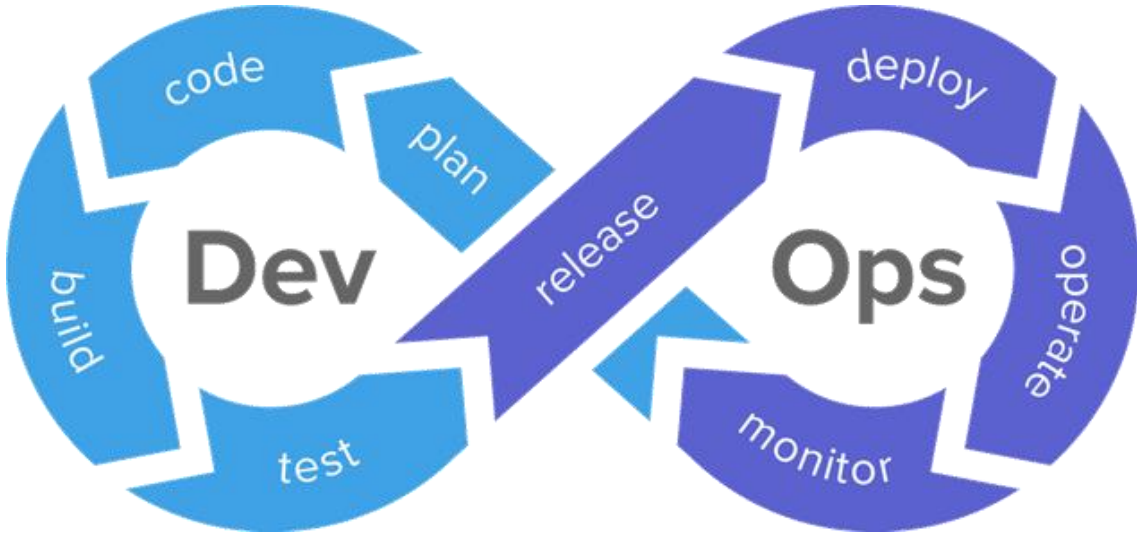


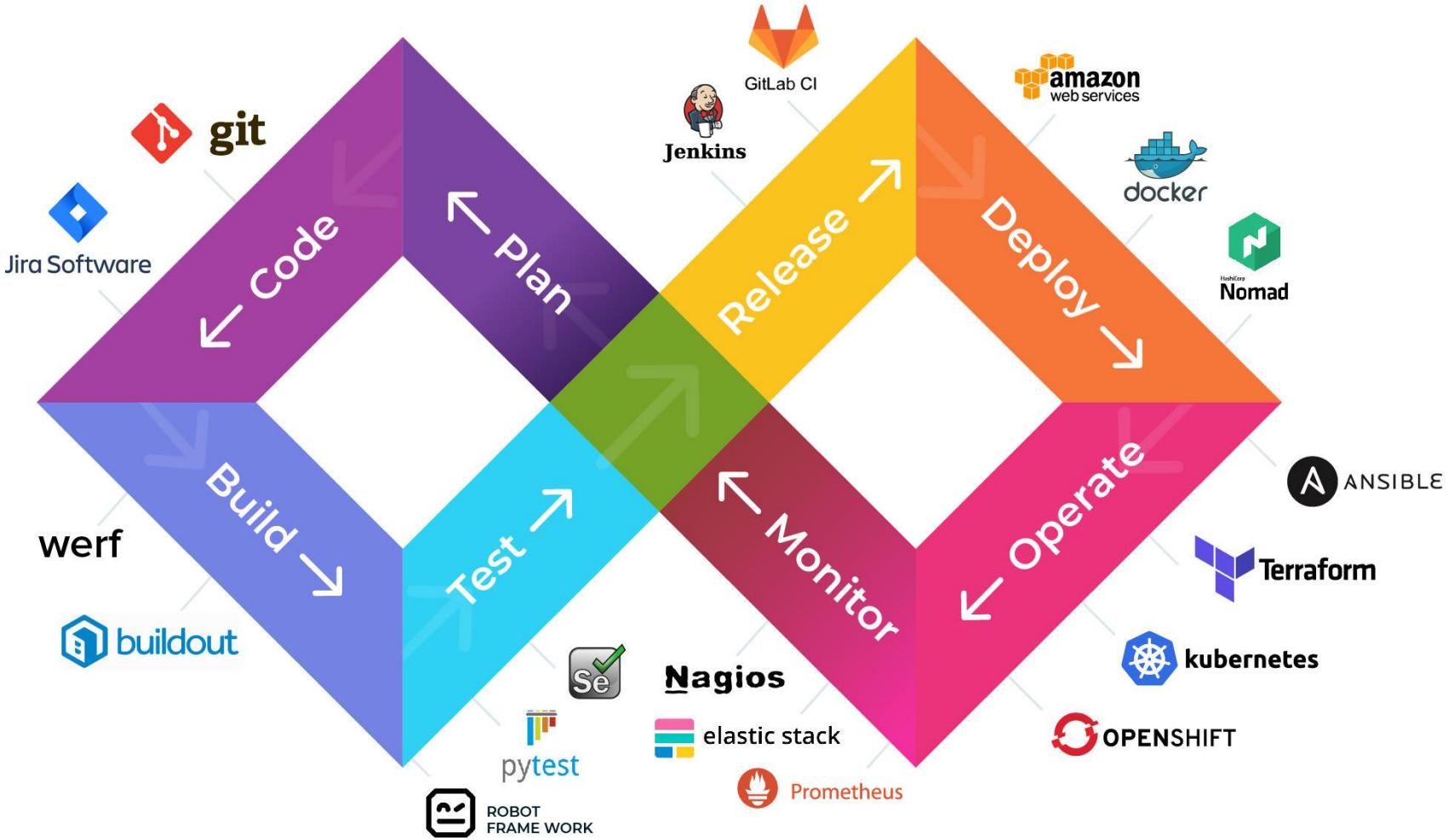
[1] J. M. Al-Kofahi, S. Kothari, and C. Kästner, “Four Languages and Lots of Macros” p. 11, 2017.

持续集成

- Continuous Integration/CI
- 当代码变动时，自动执行的系统
- 现代的持续集成工具
  - Travis CI
  - GitHub Action
  - Jenkins
  - Azure Pipelines
- GitHub Pages就是一种CI
- DevOps

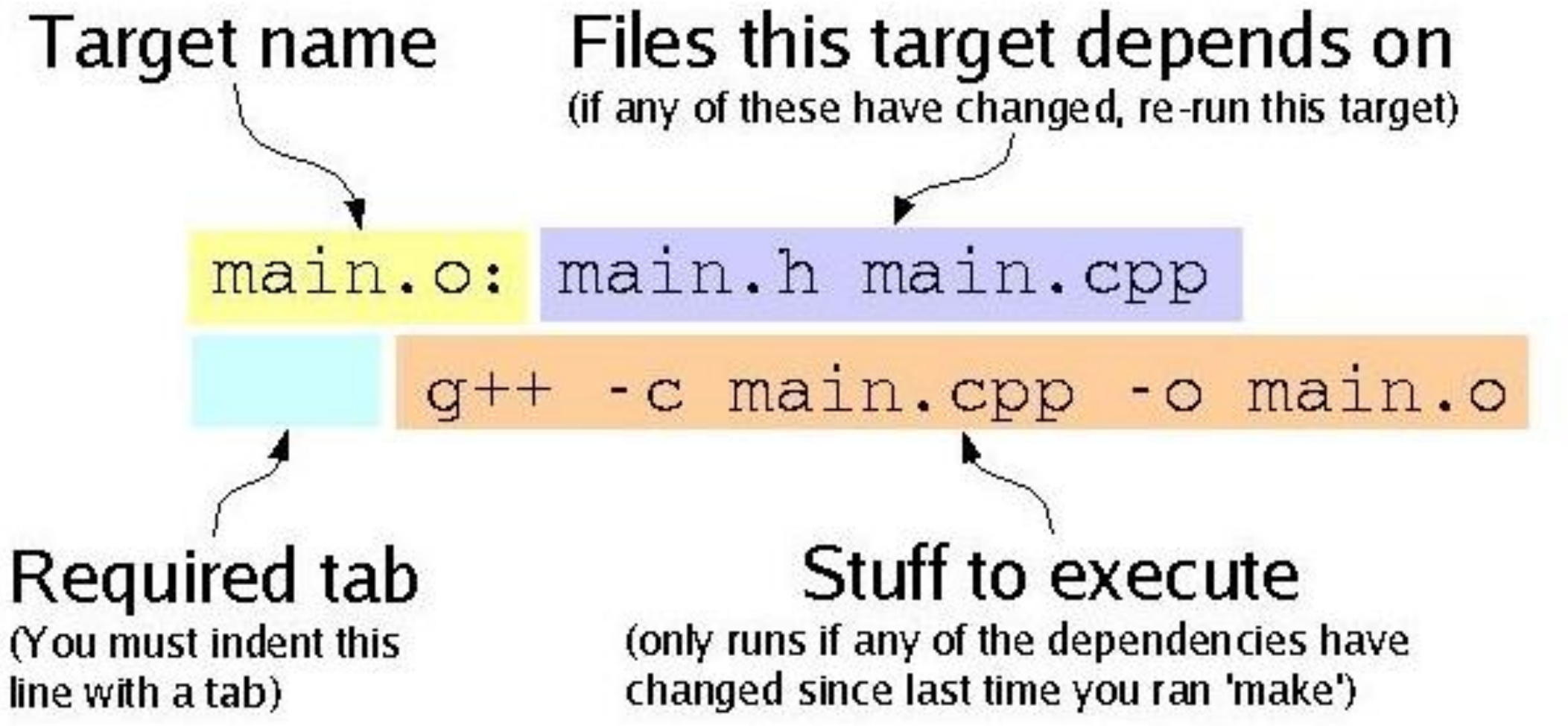

Test
Report
Continuous integration
Build
Developer
Source control
Release

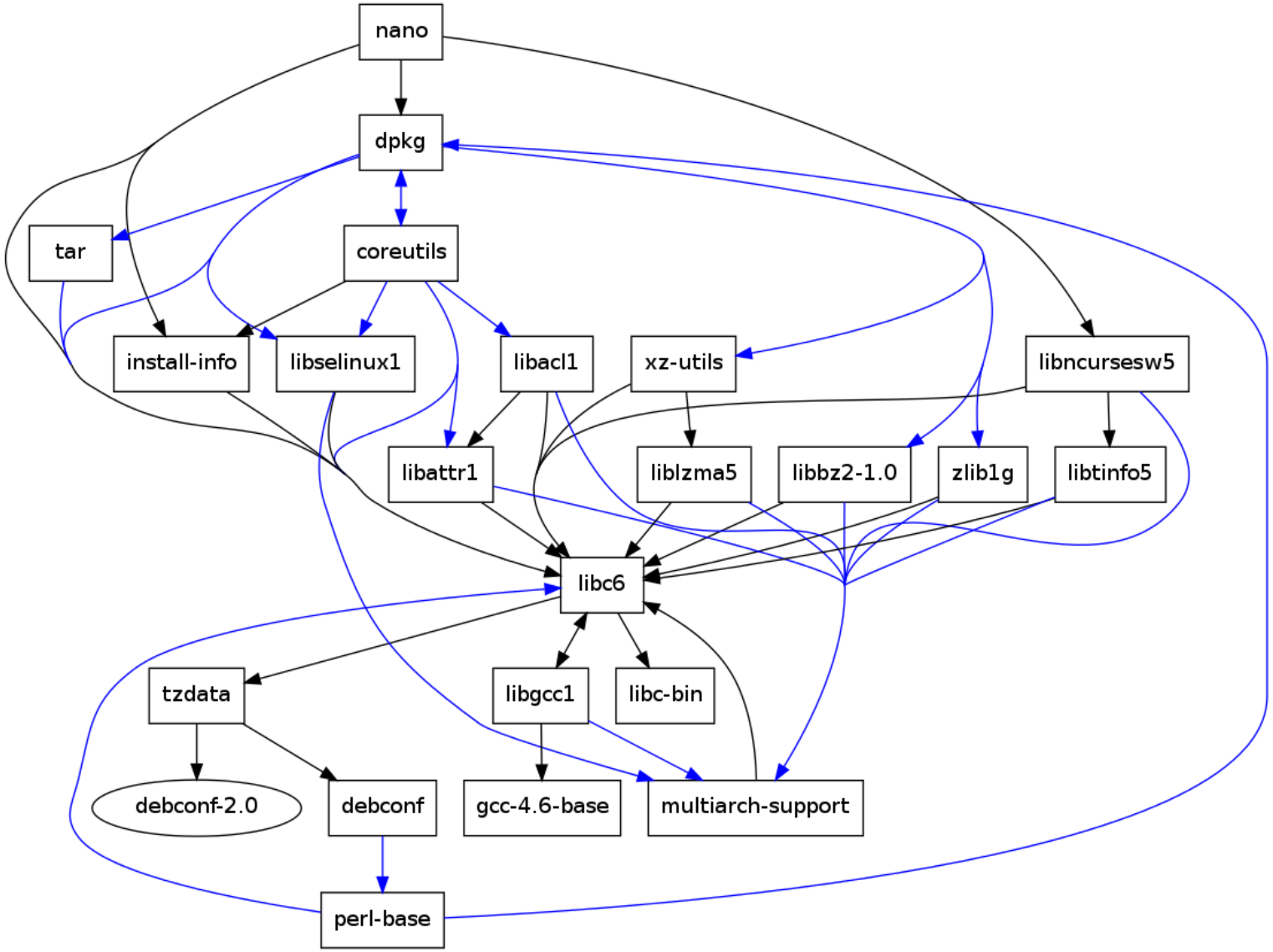

code
plan
build
test
release
deploy
operate
monitor
Dev
Ops

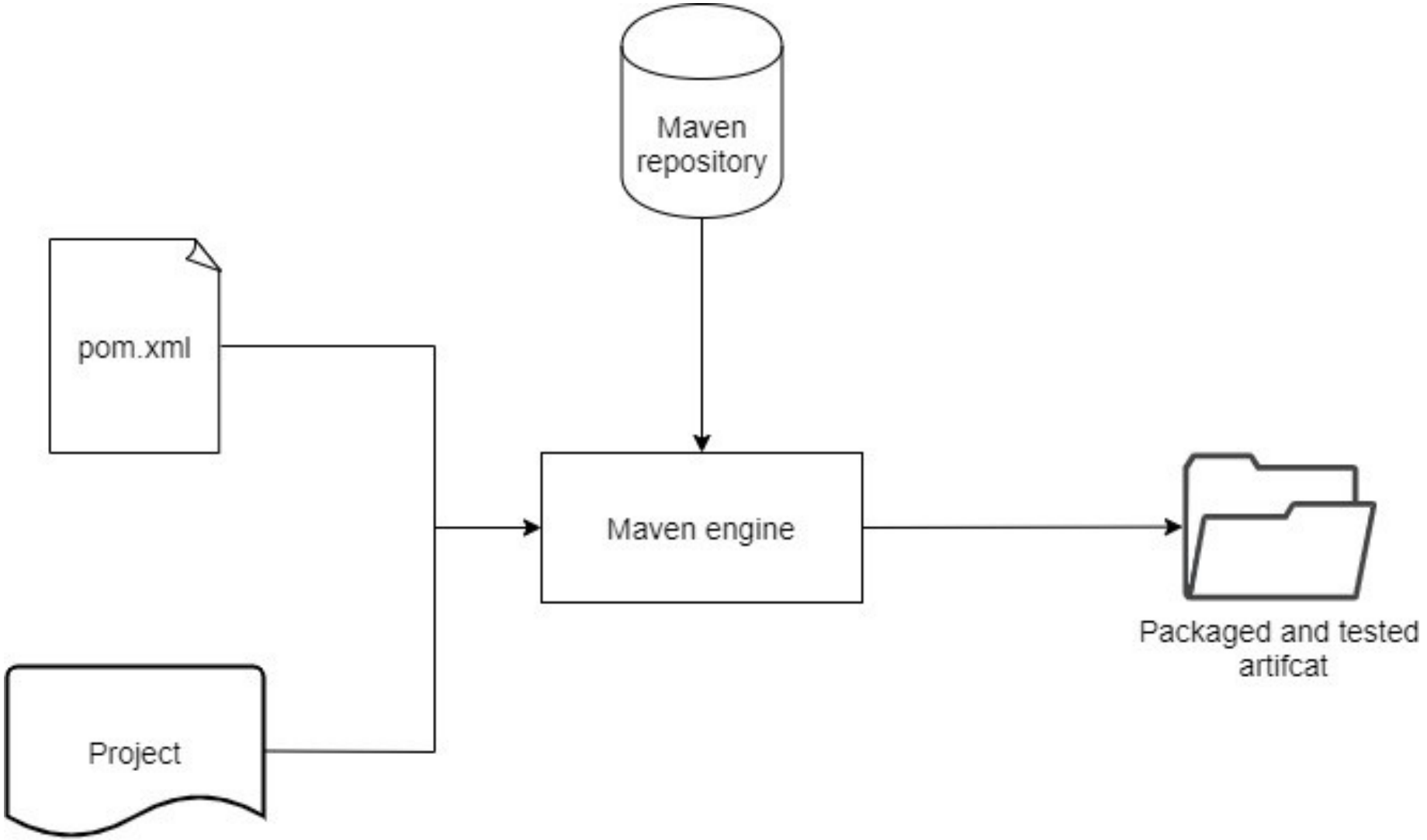

Code
Plan
Release
Deploy
Build
Test
Monitor
Operate
git
Jira Software
GitLab CI
Jenkins
amazon web services
docker
Nomad
ANSIBLE
Terraform
kubernetes
OPENSHIFT
werf
buildout
Nagios
elastic stack
Prometheus
pytest
ROBOT FRAME WORK

依赖

- 搬轮子的世界
- 依赖管理
- 不同环境的依赖管理
  - Makefile
  - APT
  - Java
  - Python


Target name
Files this target depends on
(if any of these have changed, re-run this target)
main.o: main.h main.cpp
g++ -c main.cpp -o main.o
Required tab
(You must indent this line with a tab)
Stuff to execute
(only runs if any of the dependencies have changed since last time you ran 'make')


nano
dpkg
tar
coreutils
install-info
libselinux1
libacl1
xz-utils
libncursesw5
libattr1
liblzma5
libbz2-1.0
zlib1g
libtinfo5
libc6
tzdata
libgcc1
libc-bin
debconf-2.0
debconf
gcc-4.6-base
multiarch-support
perl-base


Maven repository
pom.xml
Maven engine
Packaged and tested artifcat
Project

今日加餐： 

- 开源社区最重要的基础设施
- Issues, Pull Requests
- CI/CD: GitHub Actions
- 使用GitHub Pages部署个人网站
- 学生福利包
- 基石：Git版本管理工具
  - clone
  - pull/push
  - add, commit

现代软件栈

固件与驱动：UEFI

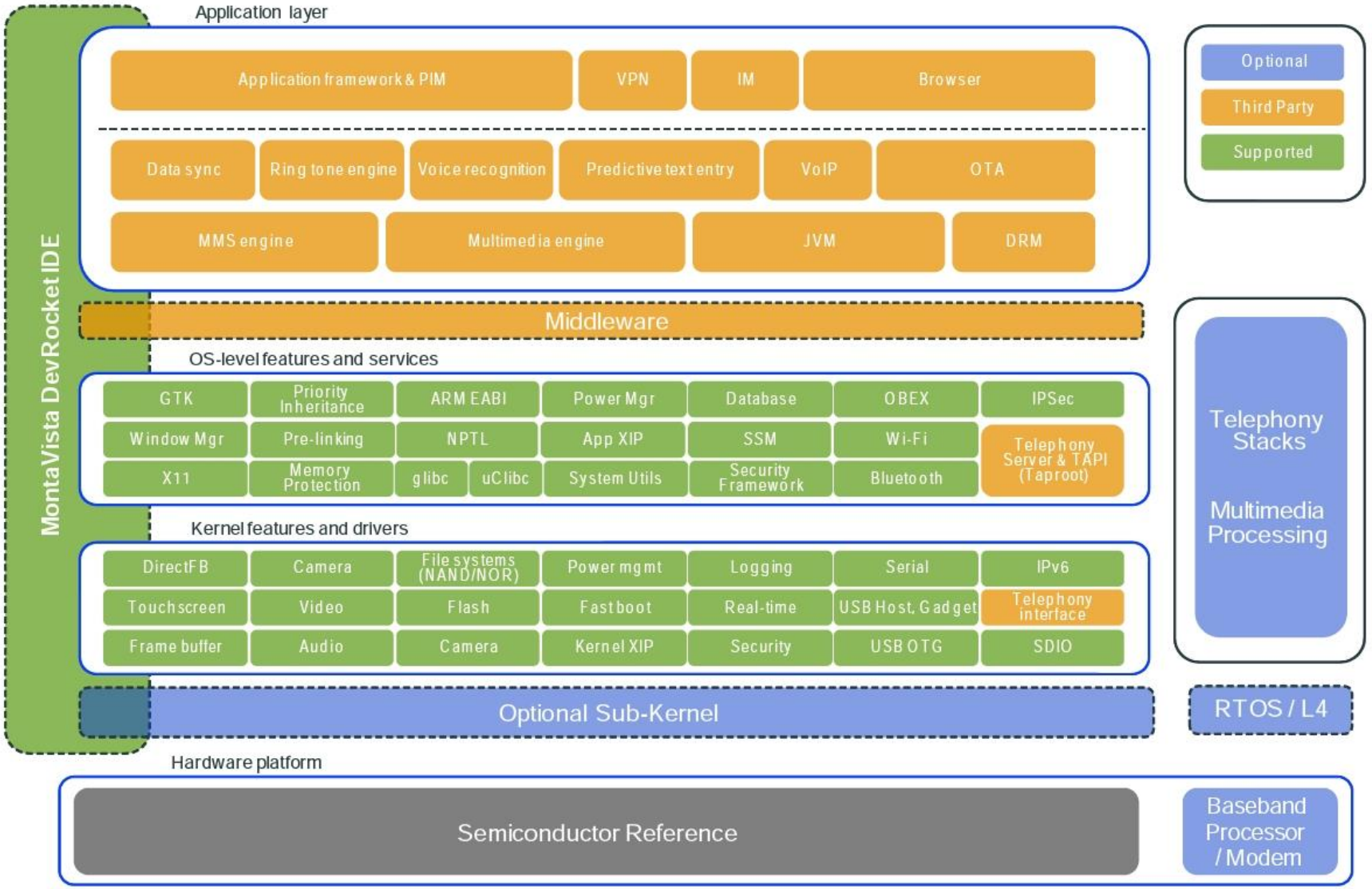
编译系统与语言标准库：GCC, GlibC...

操作系统：Linux, Windows, BSD, macOS...

中间件：网络协议栈 (TCP/IP)，数据库软件 (MySQL)，并行计算支持库 (MPI, OpenMP, CUDA)，深度学习框架(PyTorch, Tensorflow)，虚拟化软件 (VMware, VirtualBox)，GUI编程框架 (Qt, GTK)...

基础应用: 图形化界面 (GNOME, KDE)，命令行工具 (GNU coreutils)，文本编辑（vim）...

应用程序...


MontaVista DevRocket IDE
Application layer
Application framework & PIM
VPN
IM
Browser
Data sync
Ring tone engine
Voice recognition
Predictive text entry
VoIP
OTA
MMS engine
Multimedia engine
JVM
DRM
Middleware
OS-level features and services
GTK
Priority Inheritance
ARM EABI
Power Mgr
Database
OBEX
IPSec
Window Mgr
Pre-linking
NPTL
App XIP
SSM
Wi-Fi
Telephony Server & TAPI (Taproot)
X11
Memory Protection
glibc
uClibc
System Utils
Security Framework
Bluetooth
Kernel features and drivers
DirectFB
Camera
File systems (NAND/NOR)
Power mgmt
Logging
Serial
IPv6
Touchscreen
Video
Flash
Fastboot
Real-time
USB Host, Gadget
Telephony interface
Frame buffer
Audio
Camera
Kernel XIP
Security
USB OTG
SDIO
Optional Sub-Kernel
Hardware platform
Semiconductor Reference
Optional
Third Party
Supported
Telephony Stacks
Multimedia Processing
RTOS / L4
Baseband Processor / Modem

写在最后

- 计算机工程，是历史长河中的积淀，是一门经验哲学，是讲不完，讲不全，讲不准的。
- 你可以忘掉之前讲的所有，记住三点：学会搬轮子，翻文档，百度。
- 你所做的工作，要有一定的意义，提前想清楚。包括但不限于：
  - 性能更快
  - 资源消耗更少
  - 编码更容易
  - 用户体验更好
  - 安全性更高
  - 部署成本更低
- 造一套真实的系统，切忌完全的“论文导向”

我们清水河见！

感谢大家的陪伴与捧场

期待未来有机会，与大家继续交流合作！