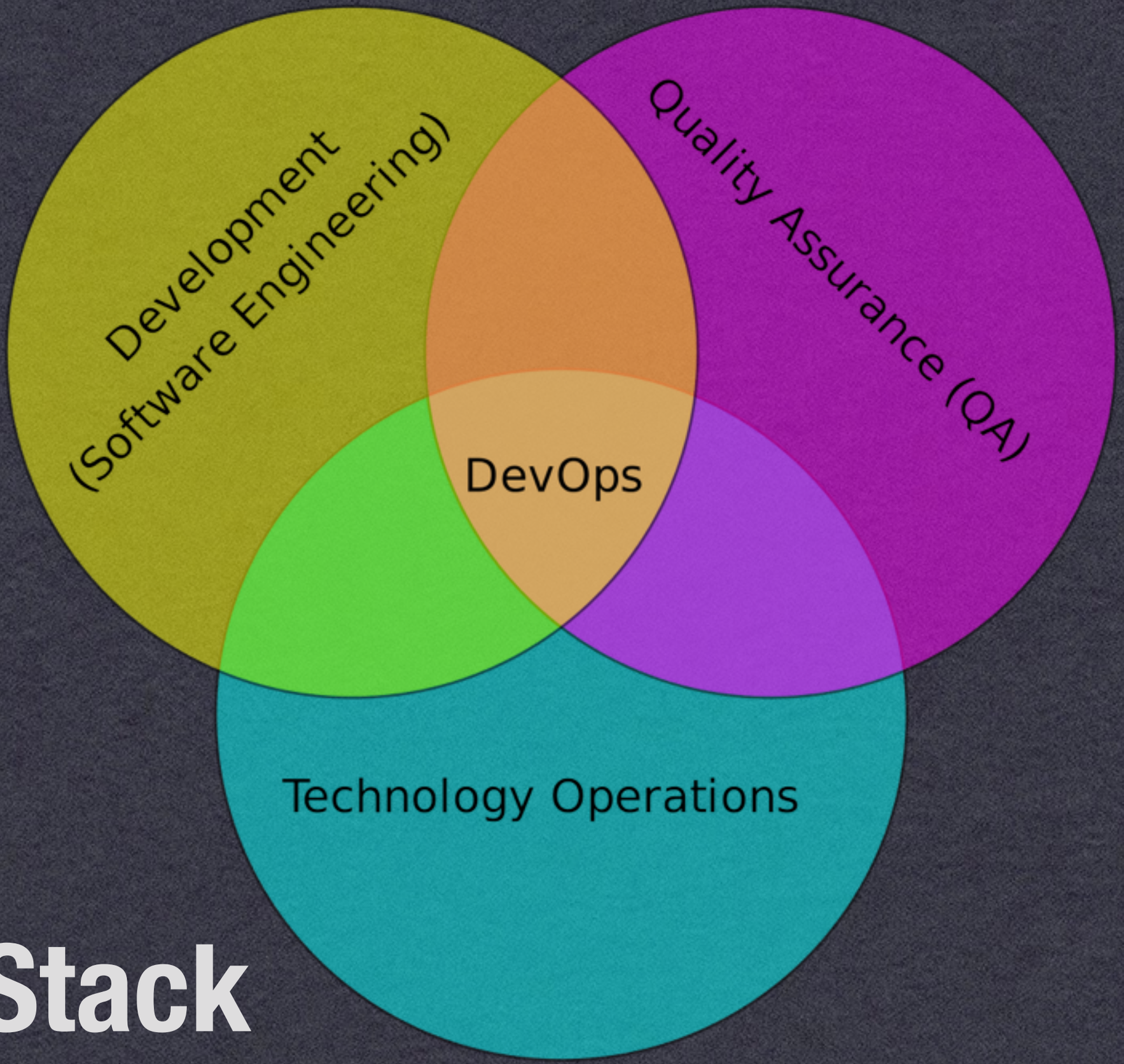
Chris Parnin: https://github.com/CSC-DevOps/Course

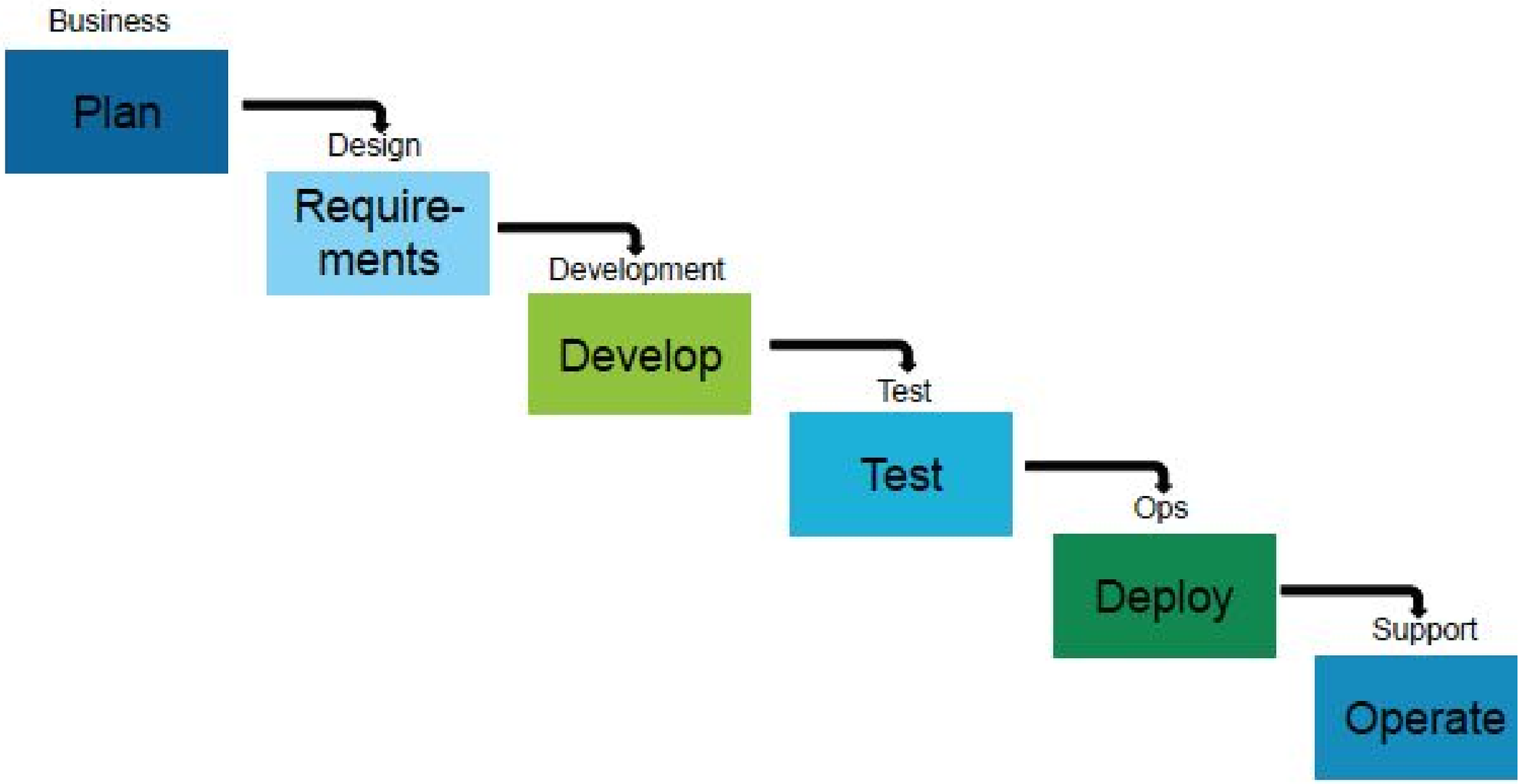


DevOps Stack

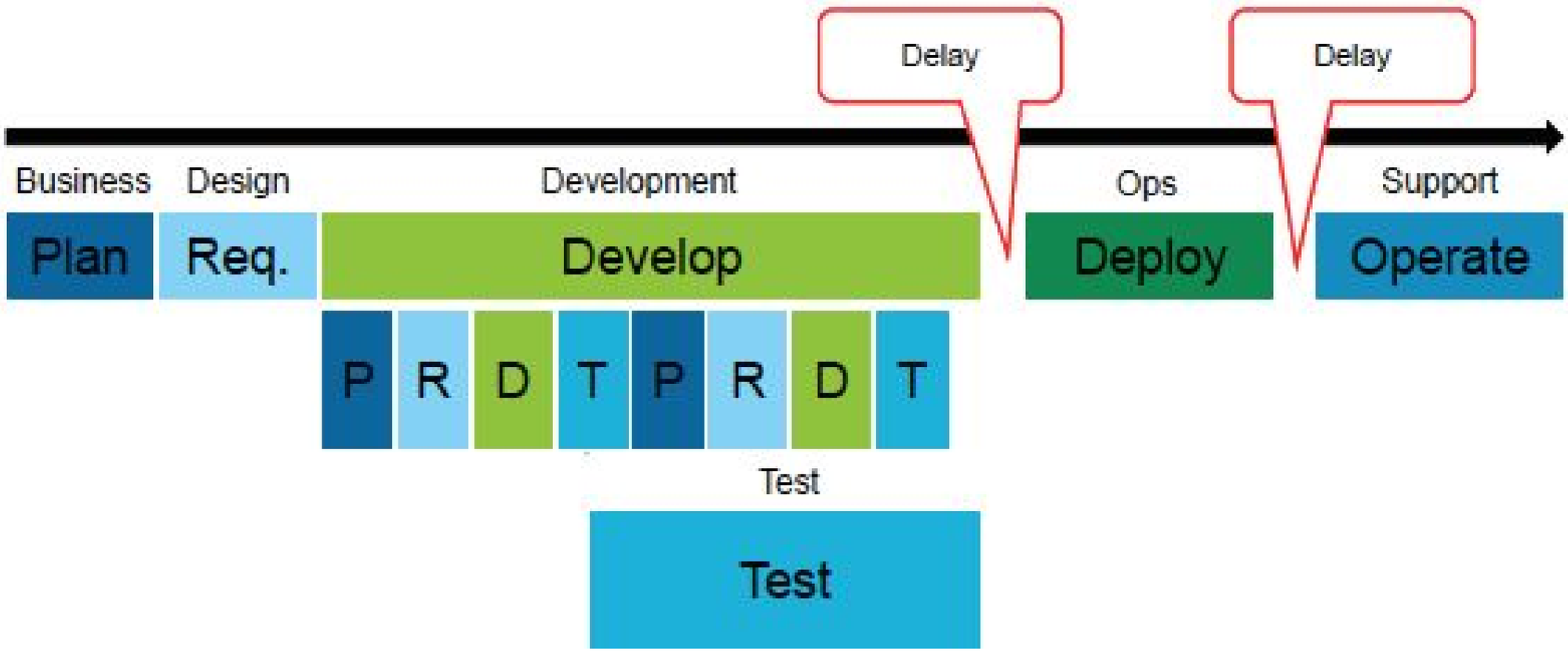
Reid Holmes

https://upload.wikimedia.org/wikipedia/commons/thumb/b/b5/Devops.svg/2000px-Devops.svg.png

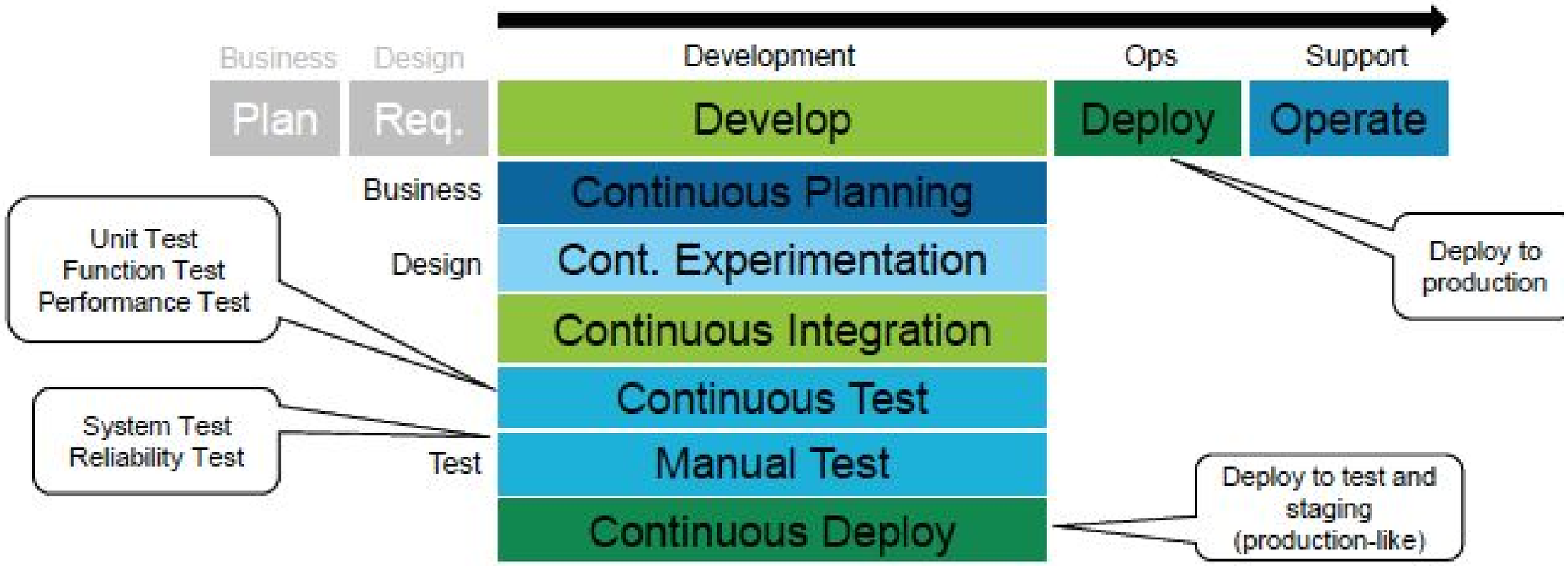
Waterfall


Business
Plan
Design
Require-
ments
Development
Develop
Test
Test
Ops
Deploy
Support
Operate

Agile


Delay
Delay
Business
Design
Development
Ops
Support
Plan
Req.
Develop
Deploy
Operate
P
R
D
T
P
R
D
T
Test
Test

Continuous Development



Who does Ops?

Who Does Operations?

Full Responsibility

Partial Responsibility

	Dev	Ops
Waterfall		Test, Staging, Production
Agile	Test	Staging, Production
DevOps	Test, Staging	Production
DistributedOps	Test, Staging, Production	Compliance and Guidance
NoOps	Test, Staging, Production	Compliance and Guidance

http://perfcap.blogspot.ca/2012/03/ops-devops-and-noops-at-netflix.html

DevOps Values

- No silos
- One team, owning changes
- Developers are responsible for supporting their code
- Leads to a quality-focused culture
- Carrying a beeper makes you careful

Configuration Management

Tracking and control activities that manage baseline and alternative versions of systems. CM enables changes to flow through different versions and releases while supporting traceability and reproducibility.

Configuration Management

- Baseline: snapshot of a system.
- All changes are submitted as change requests against the baseline.
- Gates can be established to ensure changes are scrutinized before they are applied.
- CM does not just apply to code:
  - Documentation
  - Environment
  - Libraries

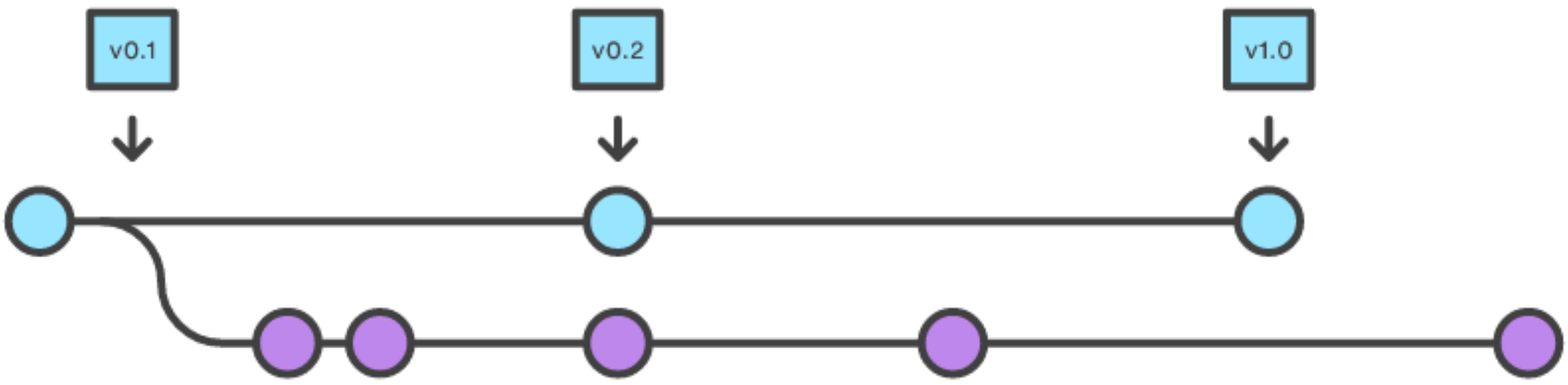
CM at Google

- Monorepo:
  - 1 billion files
  - 9 million source files
  - 2 billion LOC
  - 35 million commits
  - 45,000 commits / day
  - 86 terabytes in total

HTTPS://WWW.YOUTUBE.COM/WATCH?V=W71BTKUBDQE

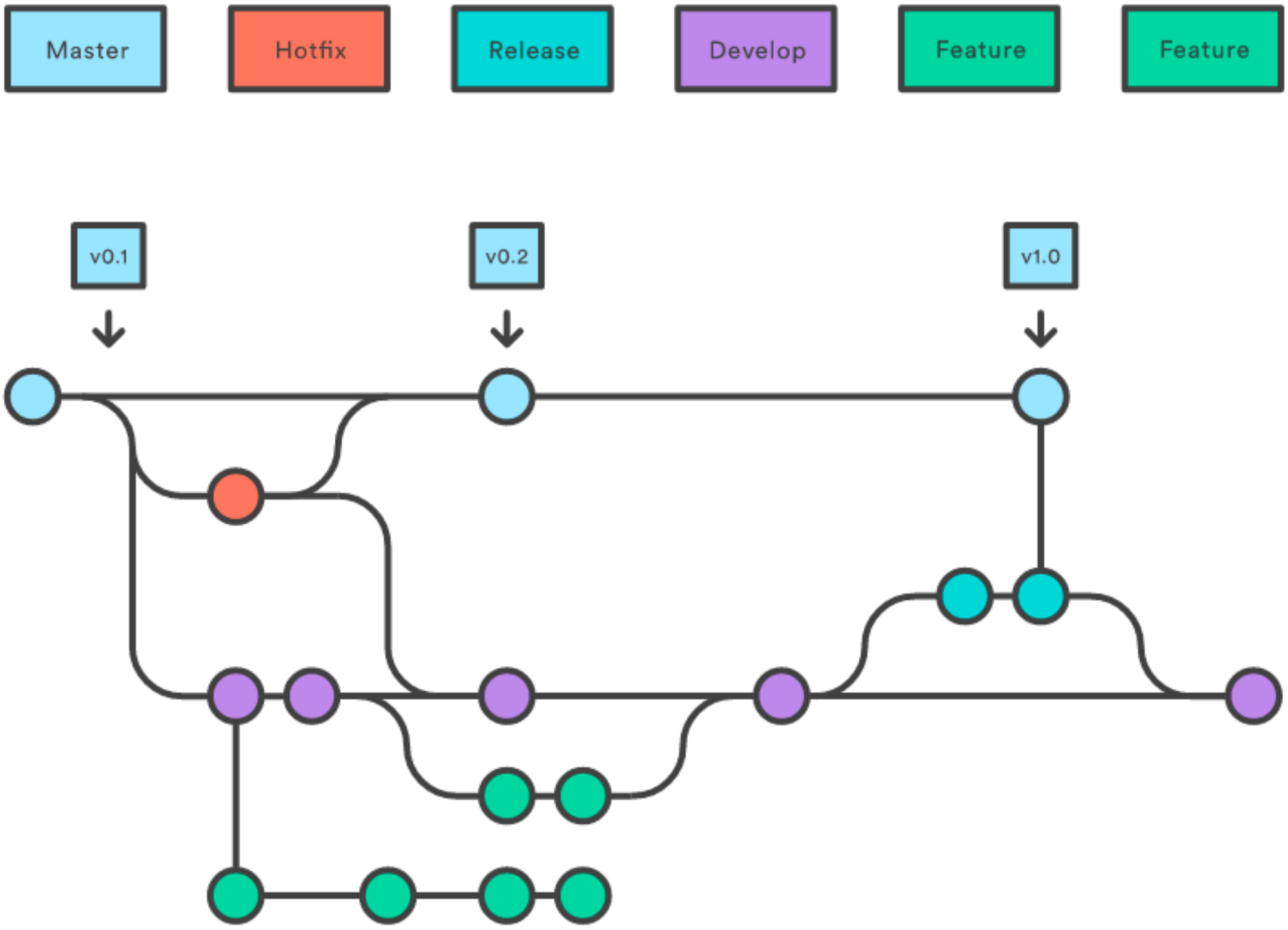
Branching

- Branches: mechanisms for allowing concurrent changes to be made to the repository.



HTTPS://WWW.ATLASSIAN.COM/GIT/TUTORIALS/COMPARING-WORKFLOWS/GITFLOW-WORKFLOW

Modern Branching

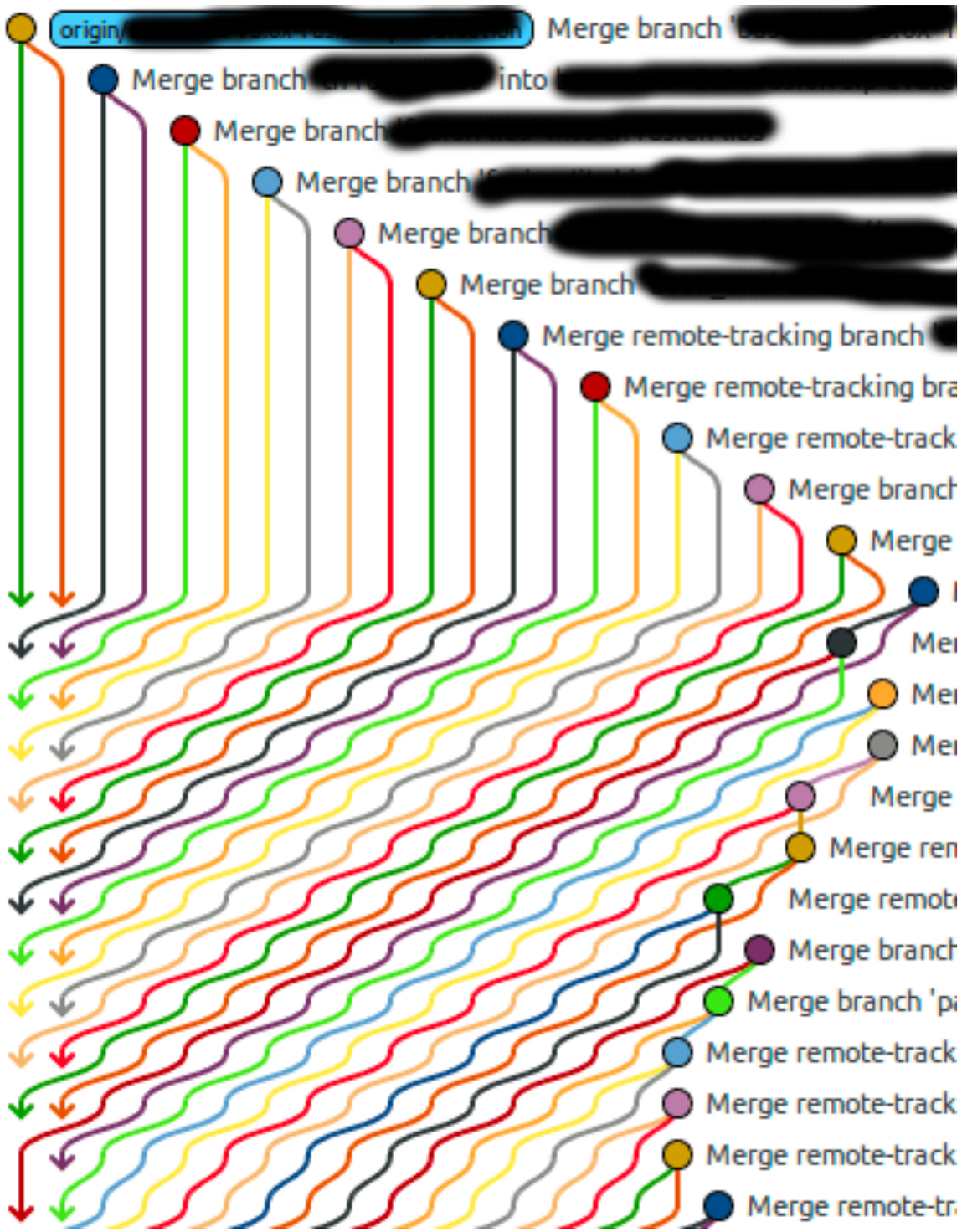


Branching Anti-Patterns

- Merge paranoia: avoid merging any changes.
- Merge mania: spend all your time merging.
- Big bang: try to merge all branches simultaneously.
- Branch mania: too many branches.
- Cascading branches: branching but never merging.
- Dev freeze: halting progress to allow merging.
- Spaghetti branching: unstructured merging.

HTTP://WWW.HILLSIDE.NET/PLOP/PLOP98/FINAL_SUBMISSIONS/P37.PDF

Branching Anti-Patterns



Nightly Build

Systems should be built and executed on a nightly basis. Smoke tests are often used to quickly check the build is not functionally broken.

Nightly Build

- Build code and run smoke tests.
  - Popularized my MSFT in the mid ‘90s
- Benefits
  - Minimizes integration risk
  - Reduces risk of low quality
  - Enables early defect diagnosis
  - Improves morale

Build Automation

- Automated build steps:
  - 1) Checkout the system.
  - 2) Acquire all dependencies.
  - 3) Build the system.
  - 4) Report build status.
- Can be built:
  - On demand
  - Through a schedule
  - Via a trigger

Build Automation

HTTPS://UPLOAD.WIKIMEDIA.ORG/WIKIPEDIA/COMMONS/9/91/BOEING-WHICHATA_B-29_ASSEMBLY_LINE_-_1944.JPG

Big Steps

1. Checkout the system:
   - Configuration Management (`git, hg`)
2. Acquire required dependencies:
   - Package Managers (`apt, npm, gem`)
3. Build the system:
   - Build Managers (`ant, mvn, gulp, rake`)
4. Run tests:
   - Test Runners (`jUnit, karma, rake test`)

Smoke Tests

- Preliminary tests to reveal catastrophic failures.
- Only a few tests will run.
  - Tests should not be flaky.
  - Tests should be cheap to run.
    - Important for systems with huge test suites.
- Validates key functionality.
  - If they do not pass, it is not worth bothering with the full suite; the build is ‘broken’.

Continuous Integration

A practice where developers automatically build, test, and analyze each change committed to the source repository.

Why CI?

- Detect and fix problems faster.
- Measurable software health.
- Document environmental assumptions.
- Effort on automation eases release stress / errors.
- Global failure feedback. Gives developers trust in their changes.

Principles

- Invest in automated tools.
- Frequent commits.
- Only commit compiling code.
- Fix broken builds immediately.
- All tests must pass.
- Run private builds.
- Pull code only from known-good configurations.

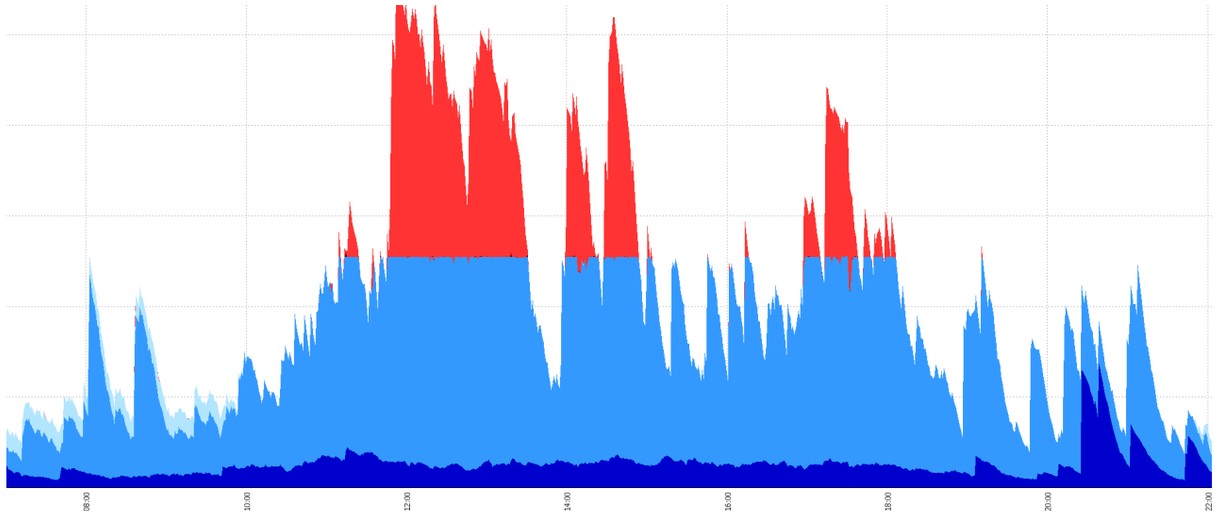
Risks (of not doing CI)

- Little confidence tests will pass after integration.
- Long integration phases.
- Inability to generate testable builds.
- Fear of making large changes (e.g., refactoring).
- Late defect discovery.
- Low quality awareness.

Automated Testing at Google

HTTP://WWW.INFOQ.COM/PRESENTATIONS/CONTINUOUS-TESTING-BUILD-CLOUD

- 10,000 devs
- 50,000 builds / day
- Monorepo
  - 20+ changes / minute
- 10 million test suites / day
  - 60 million test cases executed per day



Code Analysis

- Linting
  - Flags suspicious source code constructs.
  - Heuristic-based.
    - e.g., use before def, div/0, range violations.
      - Esp. important for interpreted languages.
- Dependency analysis
  - Checks code dependencies and identifies call sites that could be affected by a change.
- Architectural compliance checking
  - Ensure change does not violate model.


Thousands of commits per week
Total
Human
300
225
150
75
0
1/1/2011
1/1/2012
1/1/2013
1/1/2014
1/1/2015

CI Challenges

- Granularity:
  - Every change? Time delta? Batching?
- How to deal with failures:
  - Notify one person? Team? Organization?
- Tool support:
  - Remote test execution (`Jenkins, Travis CI`)
  - Track build failures (`Jenkins, Travis CI`)
  - Track per-test failures (`Jenkins`)

Continuous Delivery

A practice that ensures that a software change can be delivered and ready for use by a customer in production-like environments.

Continuous Delivery

- Dogfooding
- Heavier infrastructure needs:
  - Test infrastructure.
  - Containers helpful:
    - Build & distribute virtual environment (vagrant can manage VM or containers like Docker)
- Why Continuous Delivery?
  - Can deploy at _any_ time (e.g., security fixes).
  - Even better feedback / more confidence.

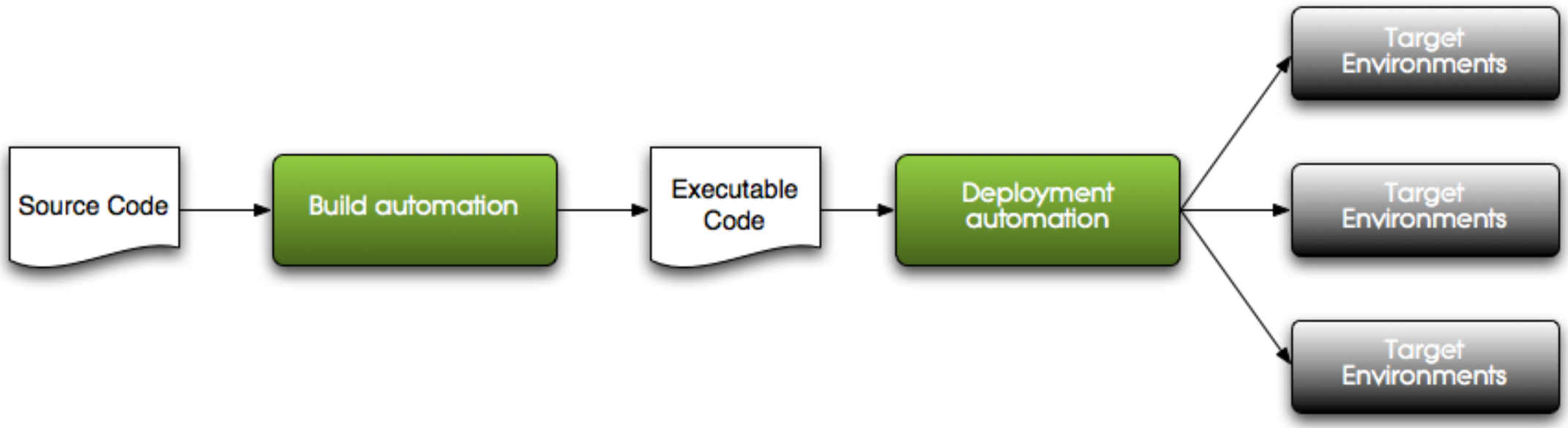
Managing Infrastructure

- Configuring hosts is not easy in practice:
- Dynamic:
  - Install/manage hosts (`cobbler`)
  - Configure host environment (`puppet, chef`)
  - Deploy code to hosts (`ansible, puppet, chef`)
  - Monitoring hosts (`puppet, AppDynamics`)
- Static:
  - Amazon Machine Image (AMI)
    - Similar to above tools but static snapshots.

Continuous Deployment

A practice where incremental software changes are automatically tested, vetted, and deployed to production environments.

Deployment Automation



HTTPS://DZONE.COM/ARTICLES/DEPLOYMENT-AUTOMATION-VS-BUILD

Example Pipeline