

Building Continuous Delivery: rock-solid builds with Gradle

David Norton
Principal Consultant
Object Partners, Inc.



I am here to put you back on schedule.

- What is Continuous Delivery?
- Pipelines
- Source control strategies
- The build phase: repeatable builds
- Demo/Next steps

→ What is Continuous Delivery?

- Pipelines
- Source control strategies
- The build phase: repeatable builds
- Demo/next steps

What is Continuous Delivery?

Continuous Delivery

“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”

<http://www.agilemanifesto.org/principles.html>

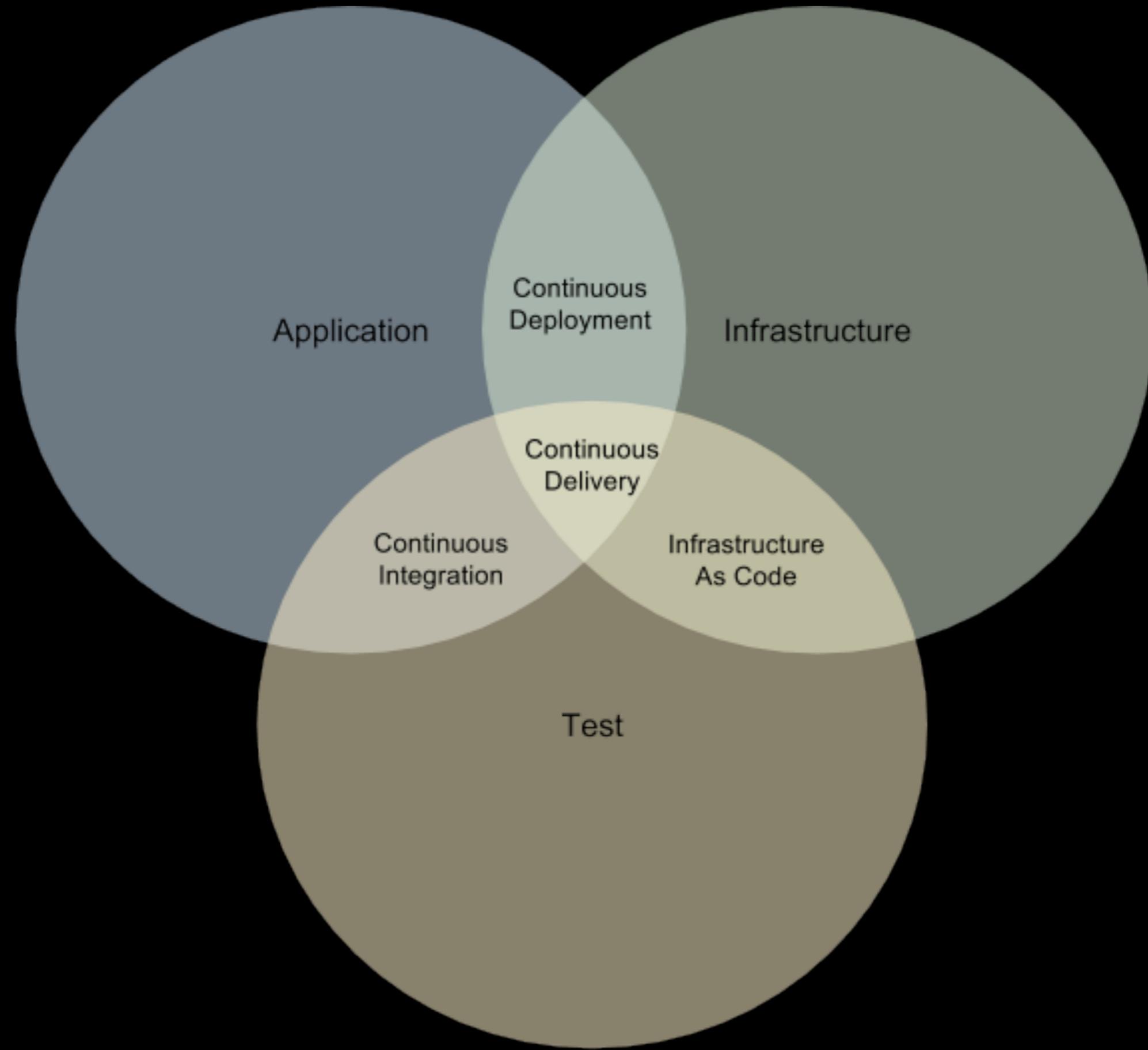
Continuous Delivery

- “Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.” - Martin Fowler

<http://martinfowler.com/bliki/ContinuousDelivery.html>

Continuous Delivery

- If you add a feature, how quickly can you get it in front of users?
- How quickly can you fix a bug? How confident are you that nothing else was broken?
- If you have an infrastructure change, how confident are you about side effects?
- If a new developer picks up the project, how easily will they be able to get started?



Continuous Delivery

- != Cowboy-coding in production
- != Continuous Deployment

Continuous Delivery

- vs. Continuous Deployment
 - Continuous Deployment can be a subset of a delivery pipeline, but rarely can/should you automate the whole pipeline to production.
 - Some applications don't lend themselves to continuous deployments (native apps, firmware, OSS...)



- What is Continuous Delivery?

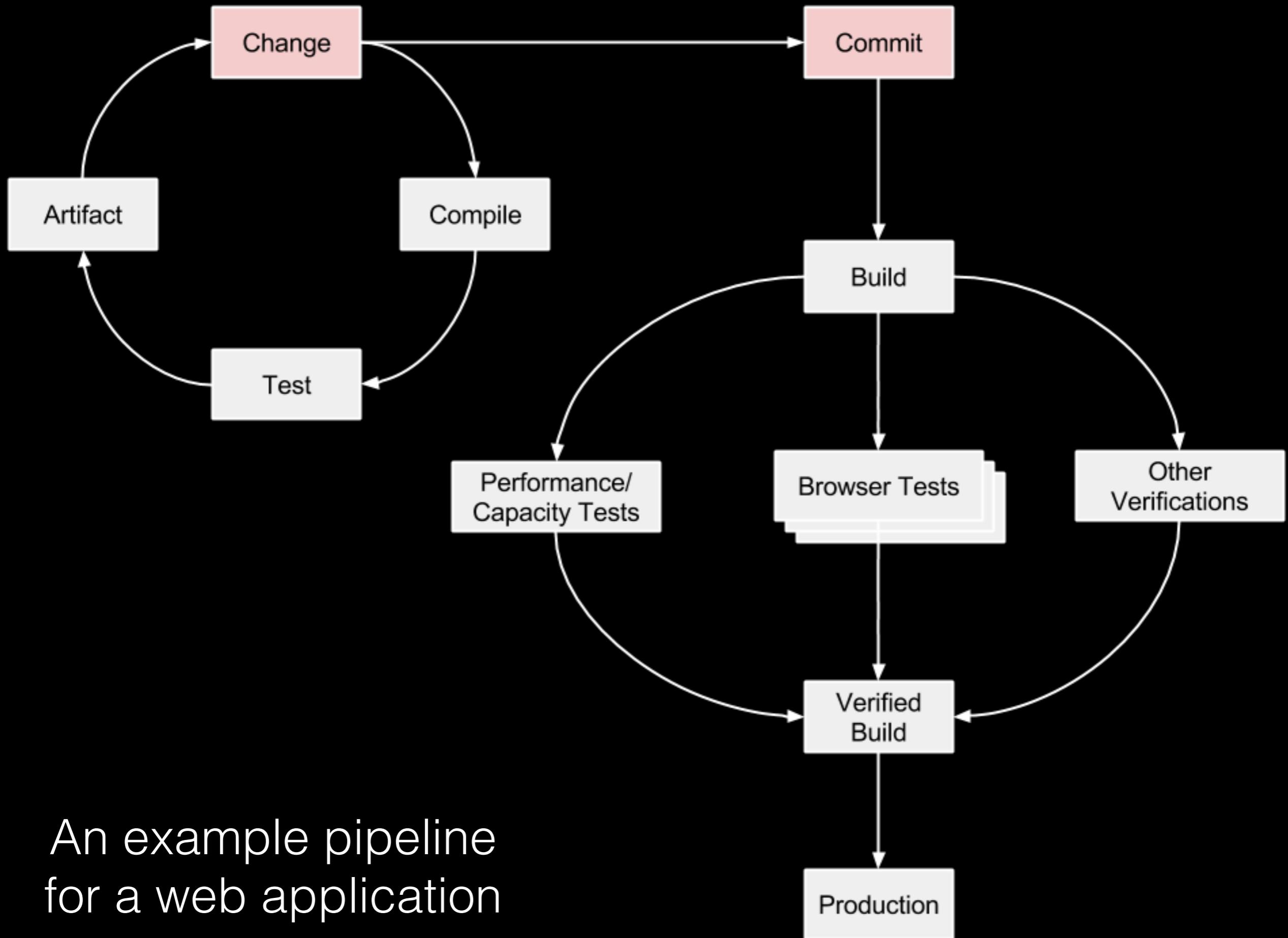
→ Pipelines

- Source control strategies
- The build phase: repeatable builds
- Demo/next steps

Pipelines

Pipelines

- One size does not fit all
- The sooner you catch a problem, the simpler it is to fix it.
- Might look something like the following:
 - Commit/Build
 - Integration / deployment to lower environment
 - Validation (acceptance, performance, code style)
 - Deployment to production



Pipelines

- Pipeline automation
 - Automate as much as possible: build, deployment, testing.
 - If necessary, some manual validation/approval checkpoints. After that, the release can continue along the pipeline.

"Works on my machine"

...says the lazy programmer

Continuous Delivery for Enterprise Peasants



But I'm just an engineer!

- You may not control the entire ecosystem...
- But you can always improve your continuous delivery practices to deliver a higher-quality product.

- What is Continuous Delivery?
 - Pipelines
- **Source control strategies**
- The build phase: repeatable builds
 - Demo/next steps

Source Control

Source Control

- Everything necessary to build project is in source control (or retrievable in a repeatable way)
- Use a README.md
- Master-based development

Master-Based Development

- Avoid long-lived feature branches
 - (Avoid Git Flow)
- Separate deployment from release with feature flags
- Branch by abstraction

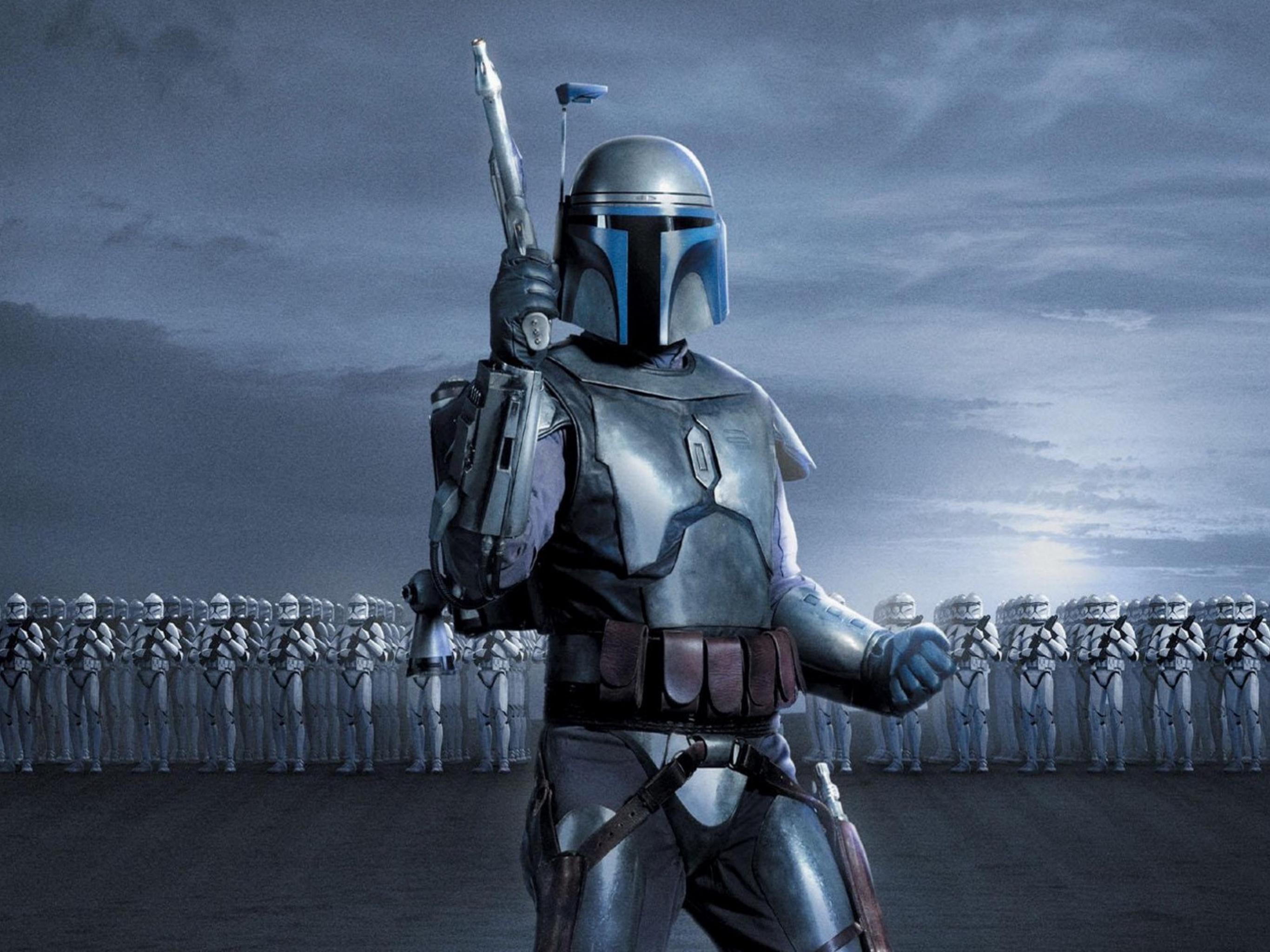
Master-Based Development

- Release branches
 - For fast-tracked fixes to releases
 - Become less necessary as cycle time decreases (automation down the pipeline improves)

Idempotent Builds

- For a given commit (input), the ability to rebuild the same artifact (output)
- Limit outside influences
- Limit side effects

- What is Continuous Delivery?
 - Pipelines
 - Source control strategies
- ⇒ **The build phase: repeatable builds**
- Demo/next steps



The build phase

The build phase

- Download dependencies
- Compile
- Unit and Integration Test
- Package
- Functional Test
- Archive Artifacts

Dependencies

Dependencies

- Avoid -SNAPSHOT, open-ended, “latest”, range dependencies. These introduce unreliable results.
- If you use a URL to load a dependency, you must be reasonably confident that resource is stable.
- Not just for libraries! Use hard versions for build tool, plugins

Dependencies

- Gradle plugin dependencies
 - Core plugins: versioned with Gradle
 - So use the Gradle wrapper!
 - Shared Build Logic - be careful
 - NOT: apply from: “<https://corpsite/my-plugin.jar>”

Compiling

Compiling

- Really there's not a lot here to screw up.
- Compiler versions, binary targets, source file encoding, etc. should be set to specific values and not use system defaults.
- Specify `sourceCompatibility/targetCompatibility` or else it will default to the current JVM.

Testing

Testing

- Unit test: Single class
 - No application framework, no database
 - Mocked dependencies

Testing

- Integration test:
 - Configured application using framework
 - Local/in-memory DB with each test rolled back via transactions
 - Mocked/stubbed external services

Testing

- Functional Testing
 - Test against application running on a running app server
 - Test APIs and/or UIs
 - More like an acceptance test

Testing

- Isolate the build from others:
 - Databases: in-memory or Docker container
 - Mock or stub third-party services
 - Betamax, MockServer
 - Stub email: GreenMail

Packaging

Packaging

- You want your build to produce an artifact that can then be deployed to every environment.
- Externalized vs. packaged config
 - API keys or other credentials might need to be externalized for security or other reasons.
 - Most configuration could be bundled into app as a named environment or profile

Archive Artifacts

Archive Artifacts

- Archive a versioned artifact that can be used for the next steps of your pipeline
- Version: specify on command line
 - e.g.: `-Pversion=1.0.$BUILD_NUMBER`
- Destination
 - Artifactory, Nexus, S3, npm, Docker Hub...

Archive Artifacts

- Gradle
 - set `rootProject.name` in `settings.gradle`. otherwise root artifact name comes from directory name

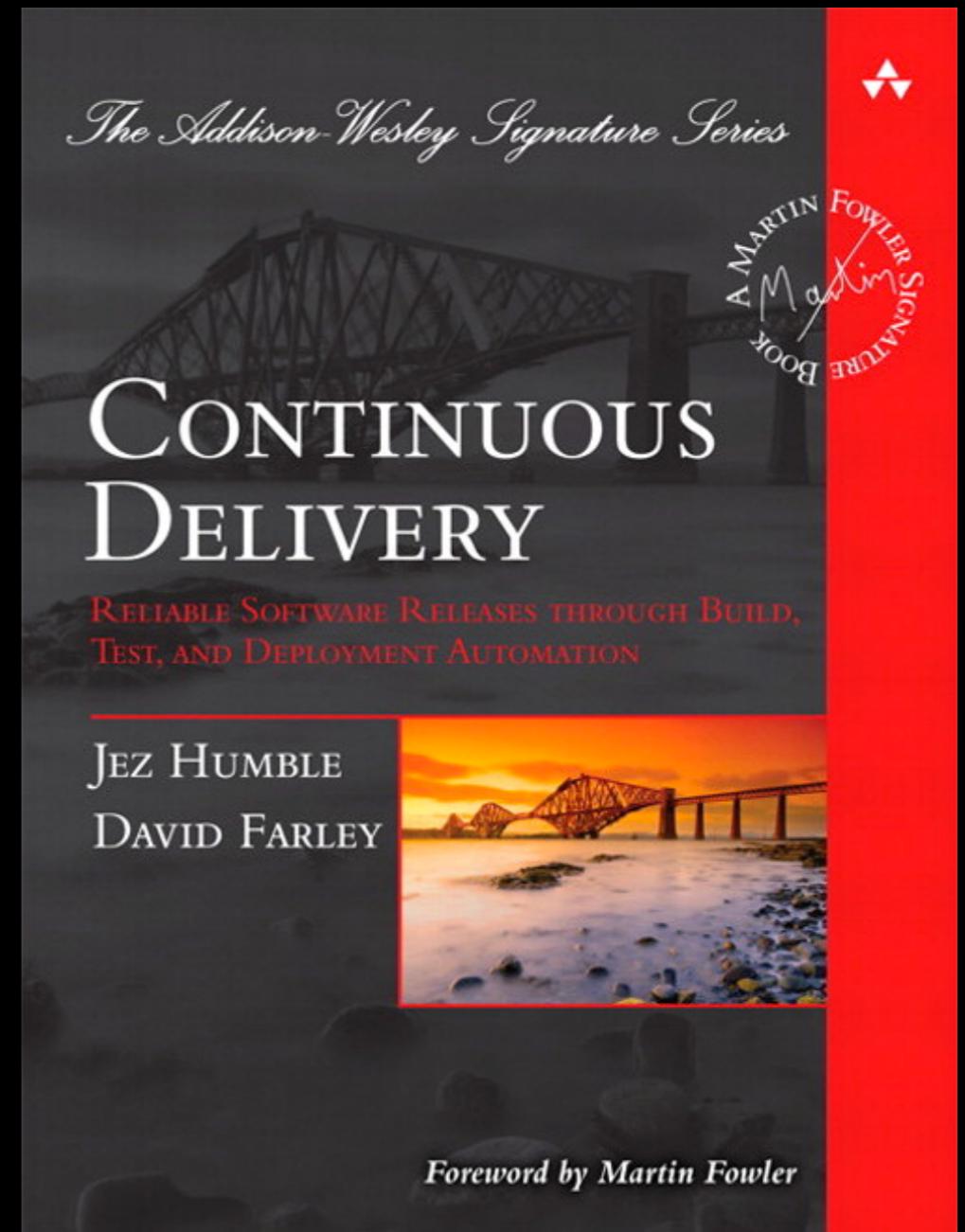
- What is Continuous Delivery?
 - Pipelines
 - Source control strategies
 - The build phase: repeatable builds
- **Demo/next steps**

Gradle Demo

Next Steps

Next Steps

- Read *Continuous Delivery* by Jez Humble and David Farley



Next Steps

- github.com/davidnortonjr/continuous-gradle
- email me, david.norton@objectpartners.com

- <http://paulhammant.com/2013/04/05/what-is-trunk-based-development/>
- [http://martinfowler.com/bliki/
BranchByAbstraction.html](http://martinfowler.com/bliki/BranchByAbstraction.html)
- <https://axelfontaine.com/blog/final-nail.html>
- Images courtesy of Disney, Wizards of the Coast, and flickr.com/photos/jdhancock/