Continuous Integration/ Continuous Delivery

Originally prepared by James Snook for CSC301

Initial Questions

- How many changes do you think you will make to your product?
- How will you make sure your changes have not broken the product?
- Who will deploy and test?
- Is it fun/best use of your time?

Late 1990s to Early 2000s

- **Testing**
  - Mostly manual
  - Performed by a separate QA team
- **Deployment & Operations**
  - Ship CD/DVD to customer, or (later) provide download link
  - Customer owns production

Late 2000s to Early 2010s

- **Testing**
  - Partly automated (esp. unit tests)
  - QA team for manual parts
- **Deployment & Operations**
  - Software as a Service
  - Deploy to in-house hardware, or third party hosted hardware
  - Vendor’s ops team owns prod

Late 2010s & 2020s (Mature Cloud)

- **Testing**
  - Fully automated
  - By dev team and/or SDETs
- **Deployment & Operations**
  - Hardware is virtual, like software
  - Infrastructure as Code (IaC)
  - Development+Operations = DevOps

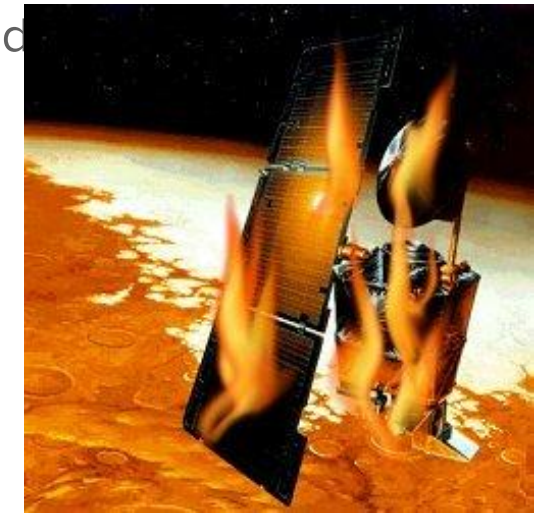
Summary of Recent History

	Testing	Deployment & Ops
Late 1990s – Early 2000s	Mostly manual	Customer installs & runs
Late 2000s – Early 2010s	QA team for manual parts	Vendor owns hardware, and installs & runs
	Automated unit tests	
Late 2010s	Fully automated	Hardware is virtual; IaC is part of codebase

QA team
Ops team
Development team… becomes **DevOps** team

1999: Mars Climate Orbiter

- Bad software can **destroy hardware**:
  - **Purpose of probe**:
    - Study the climate, atmosphere and surface changes on Mars
  - **Cost**:
    - $125 million
  - **Problem**:
    - Ground system sent commands in imperial (lbf*s)
    - Orbiter expected metric (N*s)
  - **Result**:
    - Failed orbital insertion: probe destroyed
  - **Cause:**
    - Poor integration testing

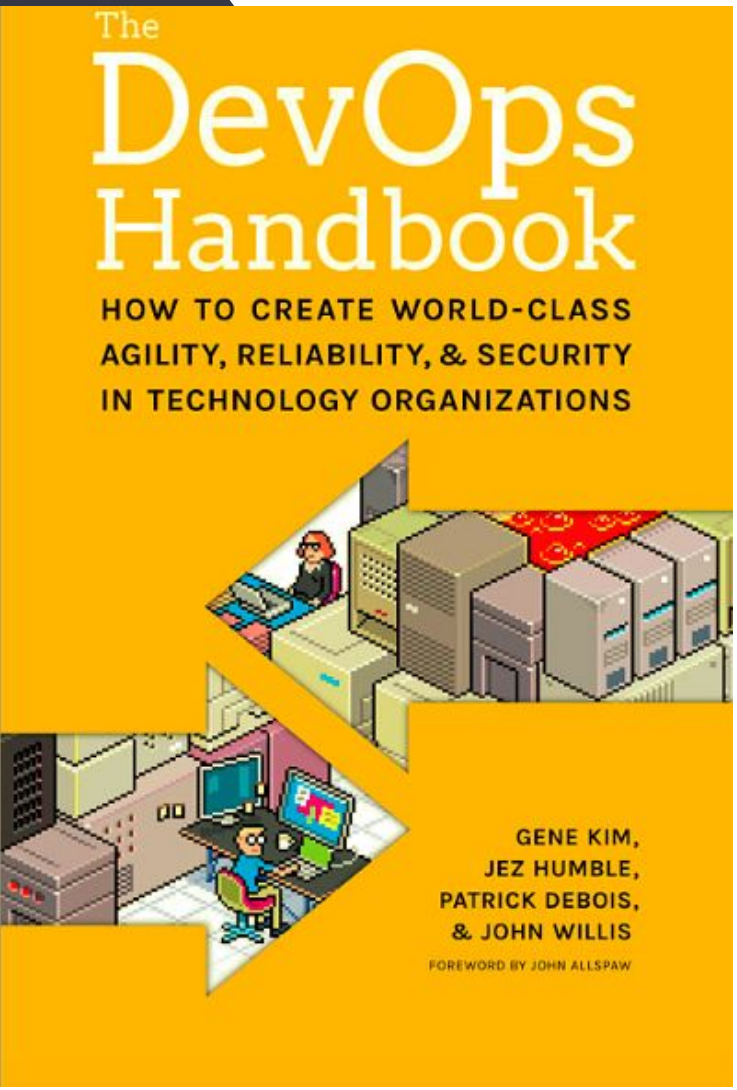


2012: Knight Capital

- Bad software can **destroy financial assets**:
  - **Knight Capital's business:**
    - Equity order routing & algorithmic trading
  - **Problem**:
    - New equity order routing code was deployed to 7 servers
    - Unfortunately, Knight had 8 production servers
  - **Result**:
    - **Knight lost $445M** and nearly went bankrupt
  - **Cause:**
    - errors during manual deployment process

THE PRESENT DAY

What is DevOps?



"DevOps and its resulting technology, architectural, and cultural practices represent a convergence of many philosophical and management movements....

...DevOps is the outcome of applying the most trusted principles from the domain of physical manufacturing and leadership to the IT value steam DevOps relies on bodies of knowledge from **Lean**, **Theory of Constraints**, the **Toyota Production System**, **reliance engineering**, **learning organizations**, **safety culture**, **human factors**, and many others."

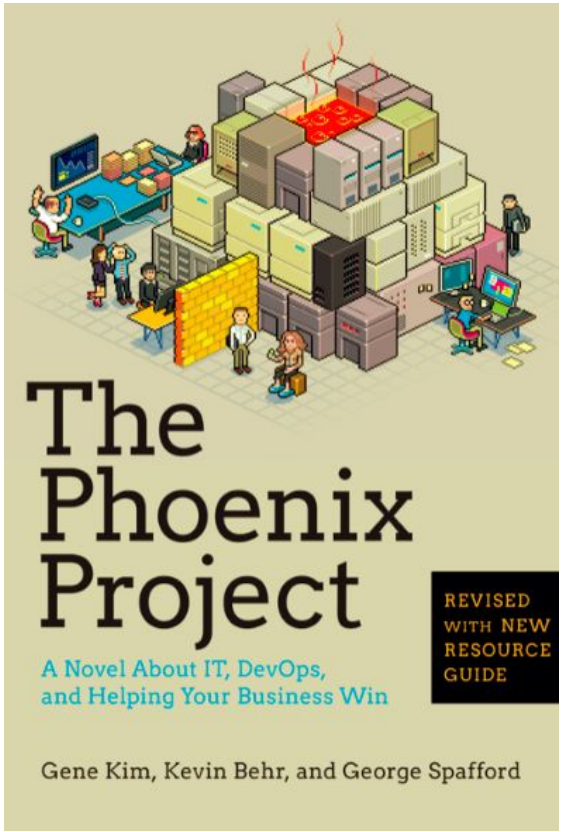
~ The DevOps Handbook

Reality: DevOps Case Study #1

"*The Phoenix Project*", predecessor to "*The DevOps Handbook*" was practically required reading within this organization.

"*The Phoenix Project*" describes the underpinning principles that all the DevOps patterns can be derived from '**The Three Ways**'.

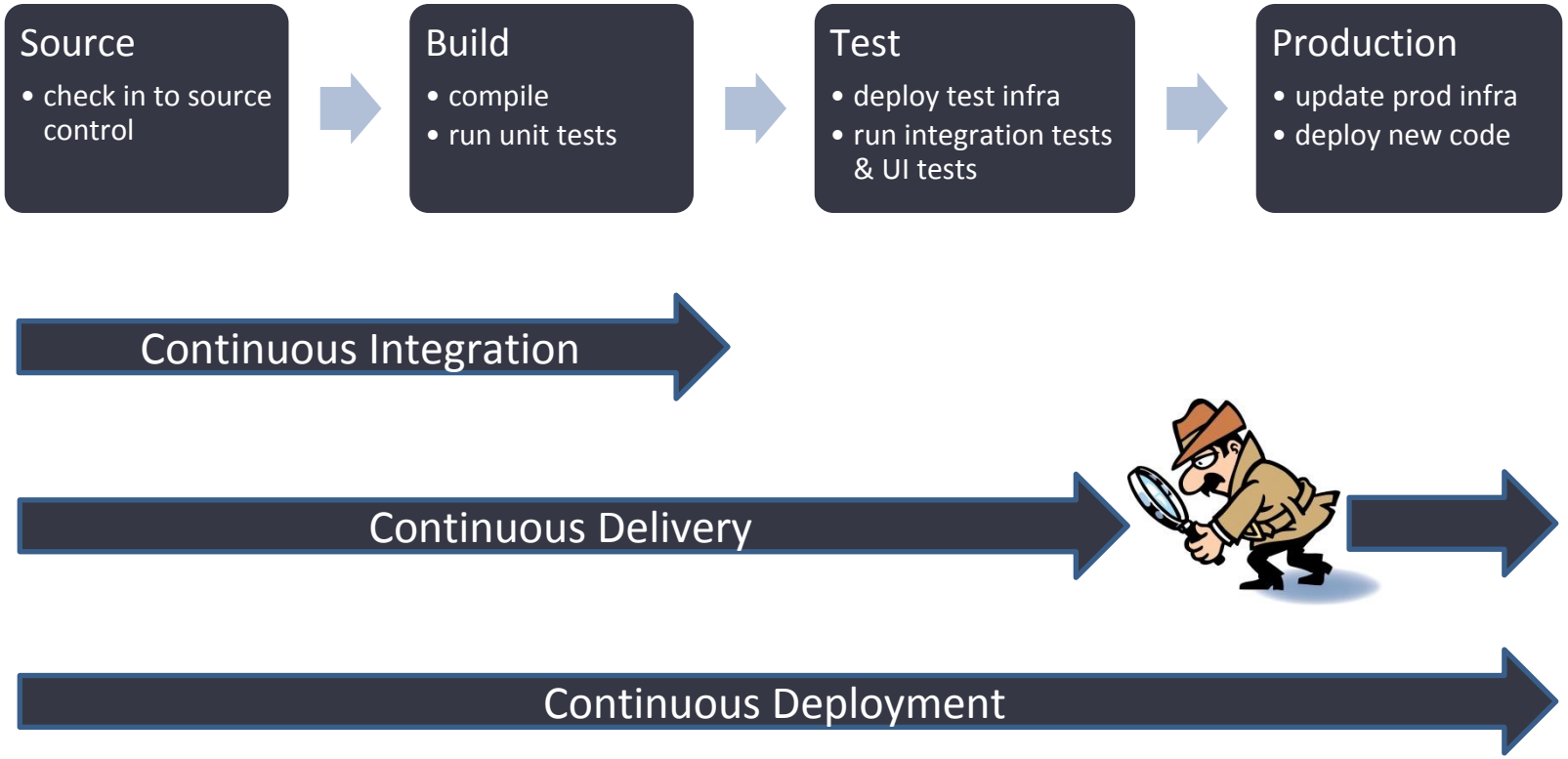
- **First Way**: left-to-right flow of work from Development to IT Operations to the customer.
  - Small batch sizes and intervals of work
  - Never passing defects to down-stream work centers
  - Constantly optimize for global goals
- **Second Way**: constant flow of fast feedback from right-to-left at all stages of the value stream,
  - Amplifying it to ensure prevention of problems from happening again or enabling faster detection and recovery.
- **Third Way**: creating a culture that fosters two things:
  - Continual experimentation, which requires taking risks and learning from success and failure
  - Understanding that repetition and practice is the prerequisite to mastery.



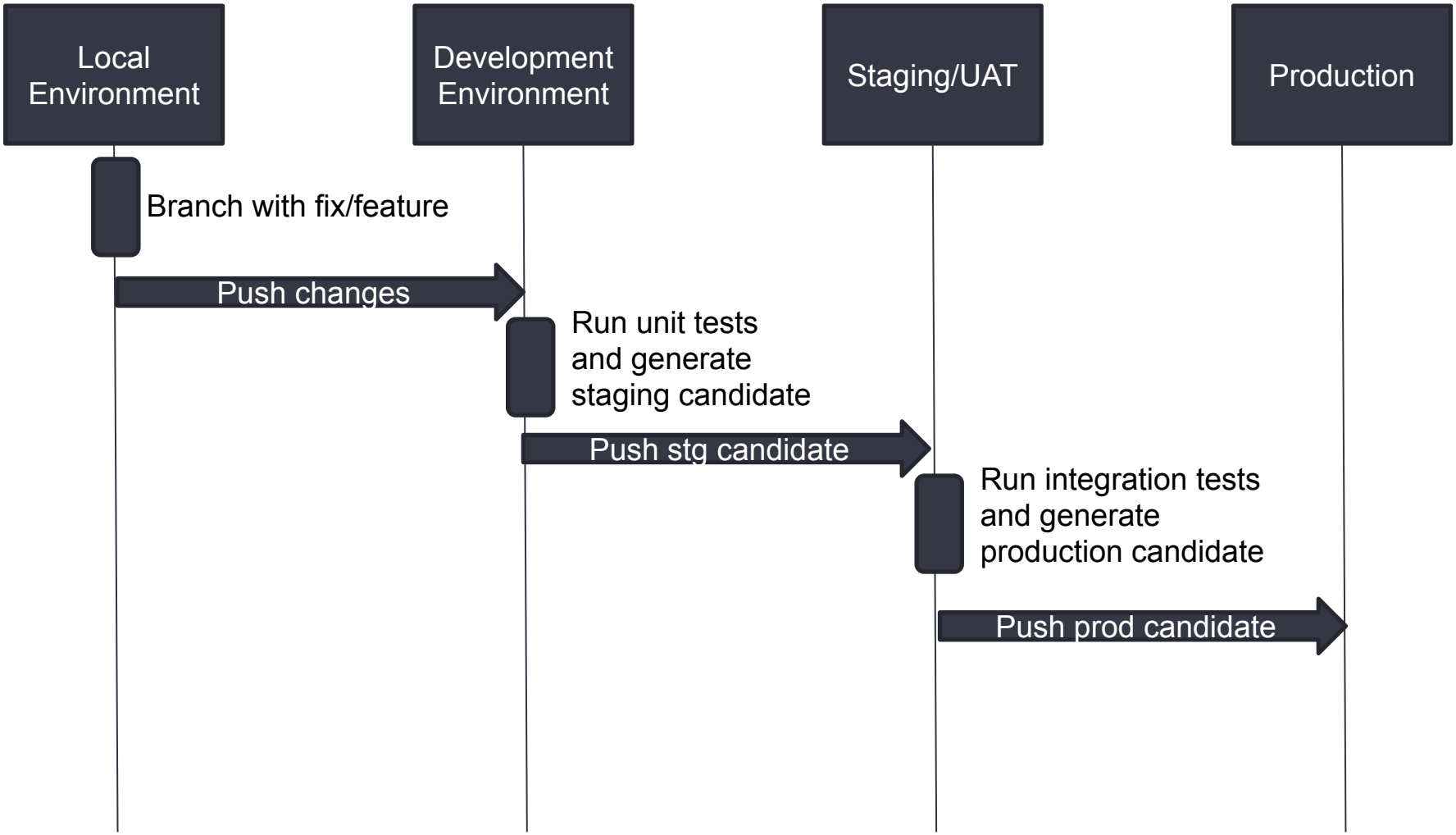
Customer Expectations

- Customers expect:
  - No installation
  - No responsibility for production operations
    - And no production downtime
  - Rapid feature delivery
  - High quality: few bugs... ideally none

How Code Gets to Production



Typical production CI/CD workflow

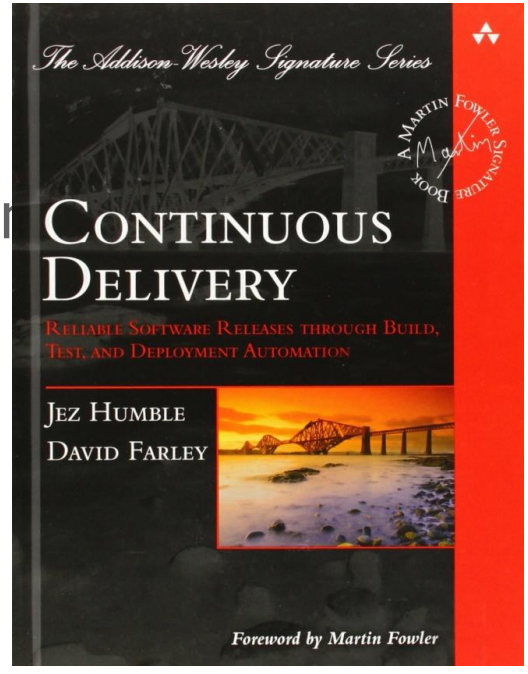


Continuous Integration

- Whenever code is submitted to source control:
  - Build it
  - Run unit tests
  - Reject commit if unit tests fail
- Why do regressive defects happen?
  - Refactoring
  - Chaos theory

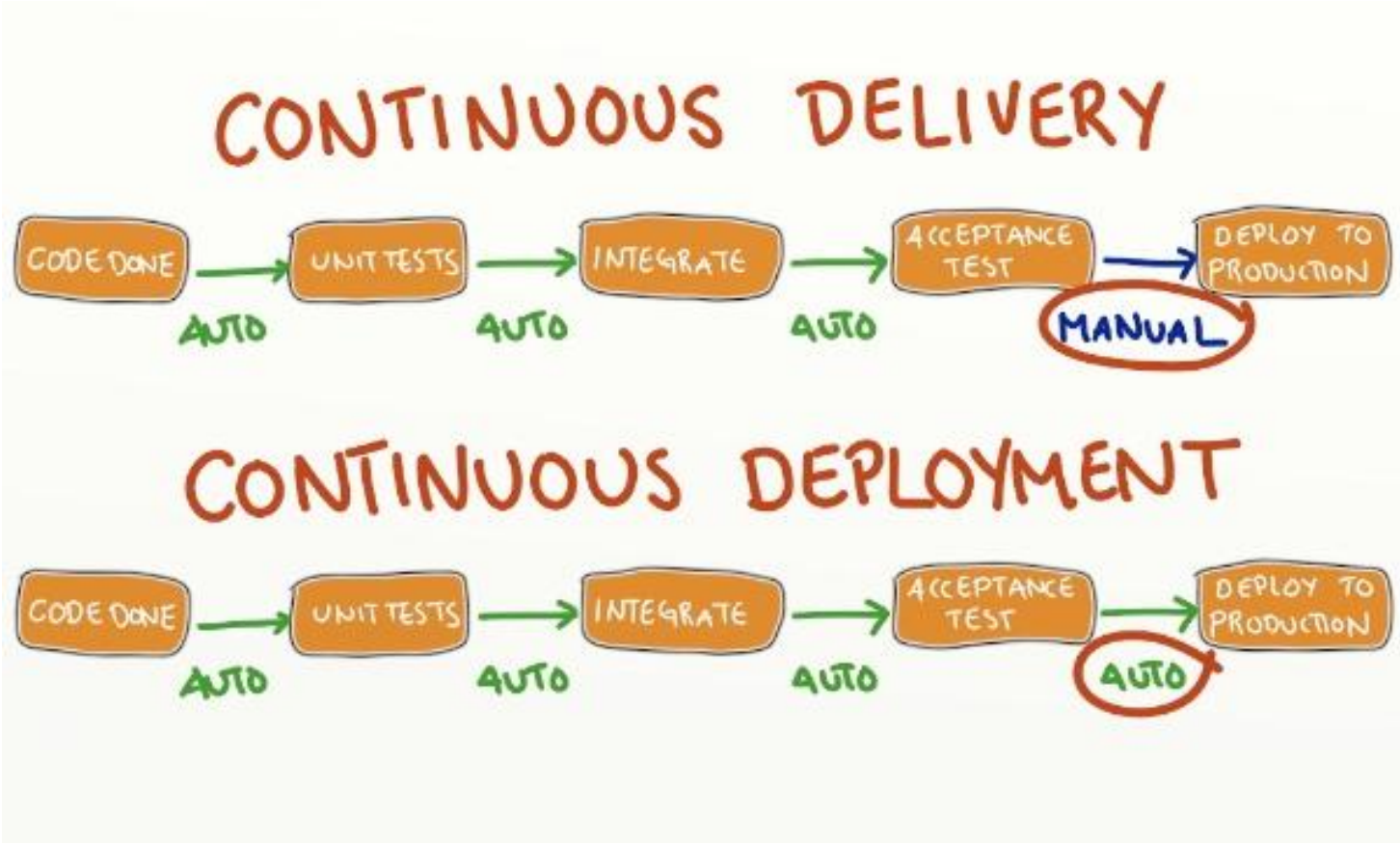
Continuous Delivery

- Continuous Integration, **plus**:
  - Deploy to Test Infrastructure
  - Run integration & UI tests
    - Against new code on new infrastructure
  - If tests pass, **notify** team that a build candidate is ready for production
  - When *release manager* is satisfied, release code and infrastructure changes to production

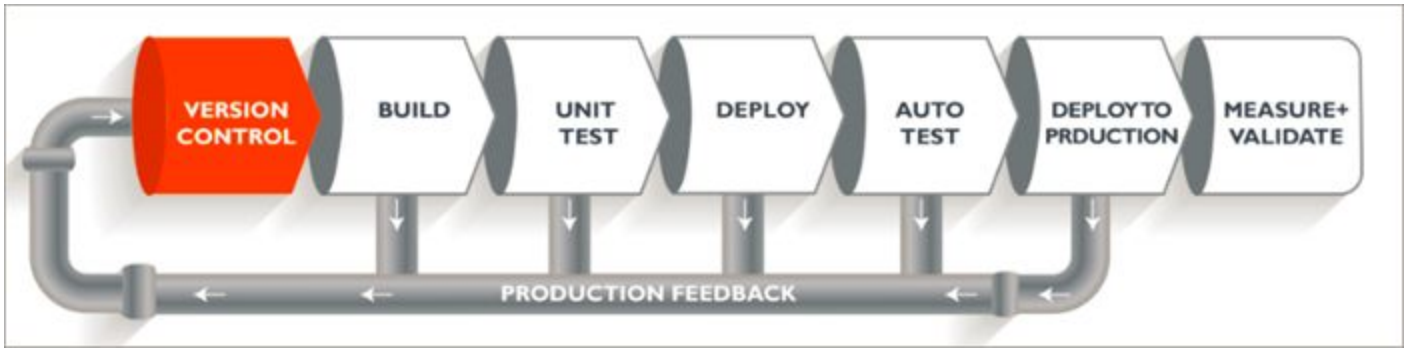


Continuous Deployment

- Continuous Delivery, **minus**:
  - Manual verification or production gatekeeping

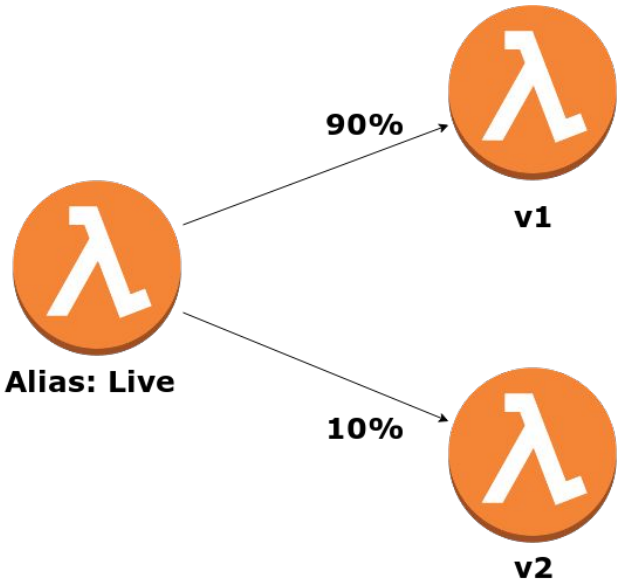


One level deeper



Finally... Testing in Production?

- It is important to be able to revert a bad build quickly, cleanly and as automatically as possible
- **Canary Deployments** and/or **Traffic Shifting** are helpful techniques
  - Can prevent major problems early on
- Initially, direct only a subset of users/requests to the new code



Production Monitoring

- So, you are the proud owner of a SaaS application in production... now what?
  - Is the application responsive?
  - Is it returning valid responses?
- You need **Monitoring & Logging**

Your Deployment Process

- How will you do your tests?
- How will you do your deployments?
    - GitHub Actions
    - CircleCI
    - Travis
    - CodeDeploy

How to configure your CI/CD

1. Track your project's metadata
   a. Package.json
   b. Requirements.txt
2. Continuously write tests (we'll talk more about it)
3. Choose your CI/CD tool and configure
   a. GitHub Actions
   b. CircleCI

Sample package.json

```
{
  "main": "node_modules/expo/AppEntry.js",
  "scripts": {
              "start": "expo start",
              "android": "expo start --android",
              "ios": "expo start --ios",
              "web": "expo start --web",
              "eject": "expo eject",
              "test": "jest --watchAll"
  },
  "jest": {
              "preset": "jest-expo"
  },
  "dependencies": {
              "@expo/vector-icons": "^10.0.0",
  },
  "devDependencies": {
              "@babel/core": "^7.0.0",
              "babel-preset-expo": "^8.0.0",
              "jest-expo": "^36.0.1"
  },
  "private": true
}
```

Summary

- Test thoroughly
- Automate your tests
- Automate your deployment process
- Dev + Test + Ops = DevOps!
- Don’t forget logging – you’ll need it in production!
- Infrastructure is part of the code

Note on teamwork

- Choose team members with complementary skills and expectations
  - Technical experience (have people with CSC343/CSC309) on your team
  - Expectations (have people who want to lead and people who want to pass)
  - Personalities (detail-oriented vs big picture, introverts and extroverts)
- Similar interests in the project