DevOps

Tejas Parikh (t.parikh@northeastern.edu)
Spring 2018

CSYE 6225
Northeastern University
https://spring2018.csye6225.com

Background

- A business must become increasingly agile to support accelerated innovation and rapidly evolving customer needs.
- Time to market is key.
- IT must become agile to facilitate business needs.
- IT operations must be able to deploy applications in a consistent, repeatable and reliable manner. This can only be achieved with the adoption of automation.

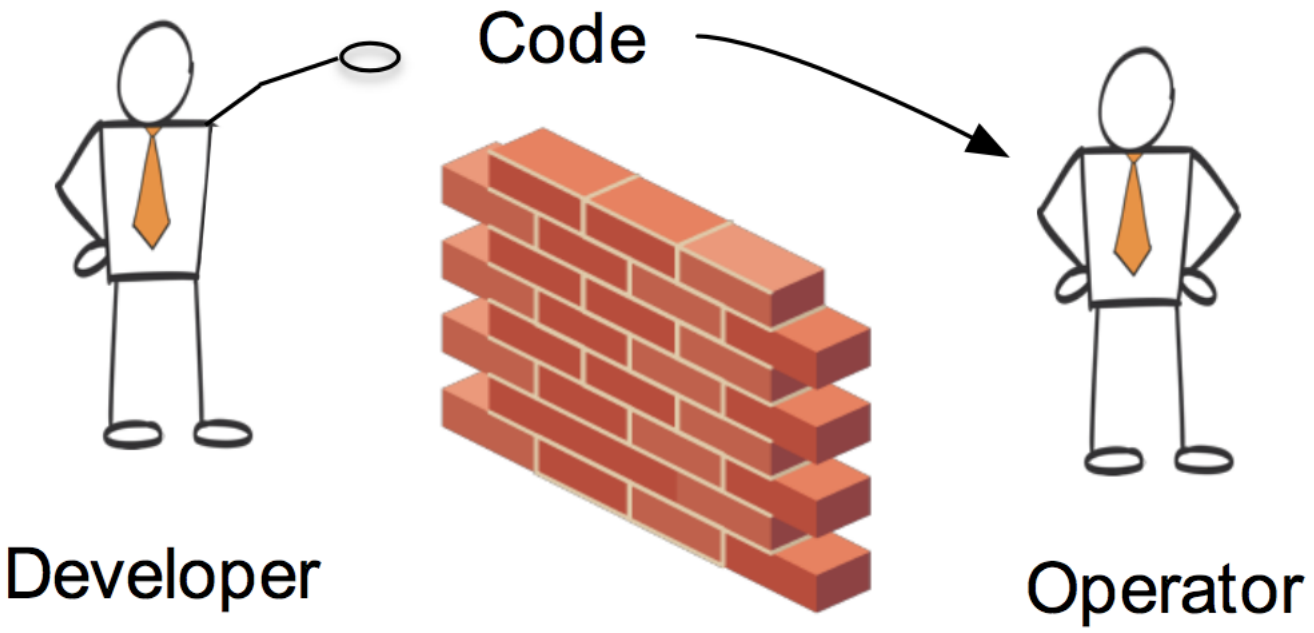
What is DevOps?

- DevOps is a new term that primarily focuses on improved collaboration, communication, and integration between software developers and IT operations.
- It's an umbrella term that some describe as a philosophy, cultural change and paradigm shift.

IT & Developer Role Merge and Follow Series of Systematic Principles

- Infrastructure as Code
- Continuous Integration
- Continuous Deployment
- Automation
- Monitoring
- Security

Traditional Deployment Model



Infrastructure as Code

- Traditionally, infrastructure is provisioned using manual process.
- A fundamental principle of DevOps is to treat infrastructure the same way developers treat code.
- Practicing "Infrastructure as Code" means applying the same rigor of application code development to infrastructure provisioning and setup.
- All infrastructure provisioning "code" and environment configuration must be stored in version management system such as Git.
- Same programming best practices must apply to the infrastructure code as applied to application code.
- Infrastructure provisioning, orchestration, and deployment should support the use of "infrastructure code".

Infrastructure as Code Technologies

- Shell/Bash Scripting
- Amazon Web Services CloudFormation

Continuous Integration

- **Continuous integration** (**CI**) is the practice of merging all developer working copies to a shared mainline several times a day.
- The main aim of CI is to prevent integration problems, referred to as "integration hell".
- CI is intended to be used in combination with automated unit tests written through the practices of test-driven development.
- In addition to automated unit tests, organizations using CI typically use a build server to implement continuous processes of applying quality control in general — small pieces of effort, applied frequently. In addition to running the unit and integration tests, such processes run additional static and dynamic tests, measure and profile performance, extract and format documentation from the source code and facilitate manual QA processes.
- This continuous application of quality control aims to improve the quality of software, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control *after* completing all development.

CI – Best Practices

- Continuous integration – the practice of frequently integrating one's new or changed code with the existing code repository – should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.
- Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build.
- The practicalities of doing this in a multi-developer environment of rapid commits are such that it is usual to trigger a short time after each commit, then to start a build when either this timer expires, or after a rather longer interval since the last build.
- Many automated tools offer this scheduling automatically.

Continuous Integration Principles

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

CI Benefits

- Bugs are detected early
- Avoids last-minute chaos at release dates
- When unit tests fail or a bug emerges, if developers need to revert the codebase to a bug-free state without debugging, only a small number of changes are lost (because integration happens frequently).
- Constant availability of a "current" build for testing, demo, or release purposes
- Frequent code check-in pushes developers to create modular, less complex code

Continuous Deployment (CD)

- Continuous deployment is another core concept in a DevOps strategy.
- CD's primary goal is to enable automated deployment of production-ready application code.
- Continuous deployment is sometimes referred to as Continuous Delivery.
- By using continuous delivery practices and tools, software can be deployed rapidly, repeatedly, and reliably. If deployment fails, it can be automatically rolled back to previous version.

Continuous Deployment (CD) Technologies

- TravisCI
- Amazon Web Services CodeDeploy

Automation

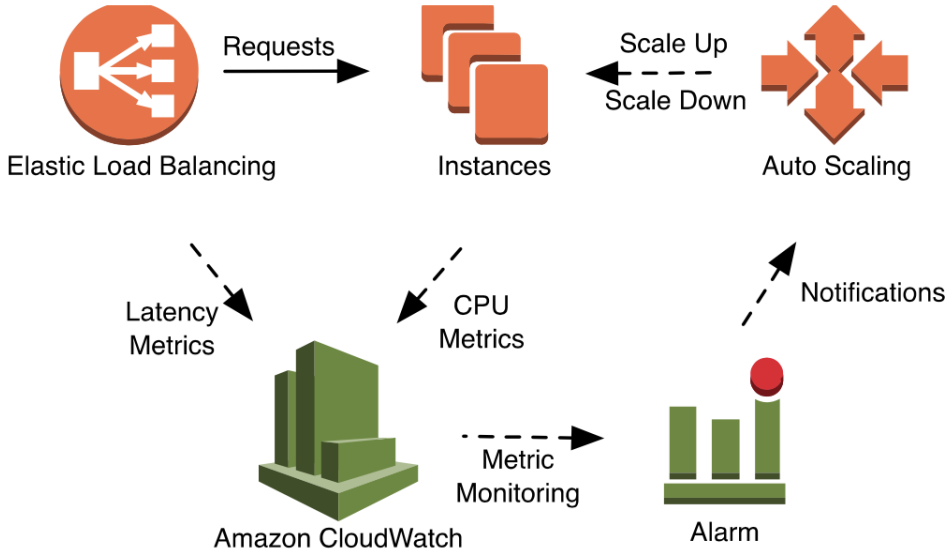
- Another core philosophy and practice of DevOps is automation.
- Automation focuses on the setup, configuration, deployment, and support of infrastructure and the applications that run on it.
- By using automation, you can set up environments more rapidly in a standardized and repeatable manner. The removal of manual process is a key to a successful DevOps strategy.
- Historically, server configuration and application deployment has been a predominantly a manual process. Environments become nonstandard, and reproducing an environment when issue arises is difficult.

Benefits of Automation

- Rapid Changes
- Improved Productivity
- Repeatable Configurations
- Leveraged Elasticity
- Leveraged auto scaling
- Automated Testing

Monitoring

- Communication and collaboration is fundamental in a DevOps strategy.
- To facilitate this, feedback is critical.
- Feedback comes from logs, monitoring, alerting and auditing infrastructure so developers and operations teams can work together closely and transparently.



Security

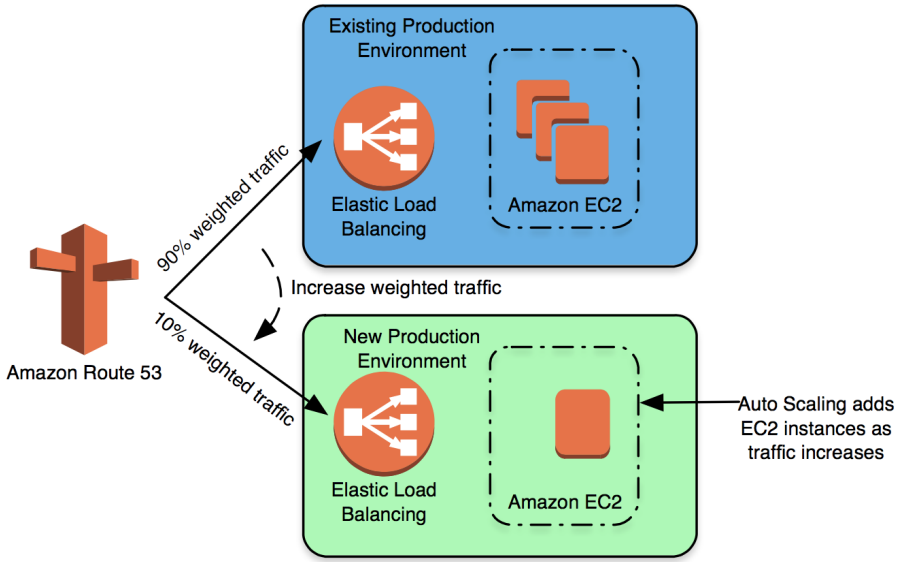
- In a DevOps enabled environment, focus on security is still of paramount importance.
- Infrastructure and company assets needs to be protected, and when issue arise they need to be rapidly and effectively addressed.

Summary

In order to make the journey to the cloud smooth, efficient and effective, technology companies should embrace DevOps principles and practices.

Blue-Green Deployment

- Blue-green deployment is a deployment practice that may uses domain name services (DNS) to make application deployment.
- The strategy involves starting with an existing (blue) environment while testing a new (green) one.
- When the new (green) environment has passed all the necessary tests and is ready to go live, you simply redirect traffic from old environment to the new one via DNS.



Additional Resources

https://spring2018.csye6225.com