Continuous Deployment with GitHub, TravisCI & Amazon CodeDeploy

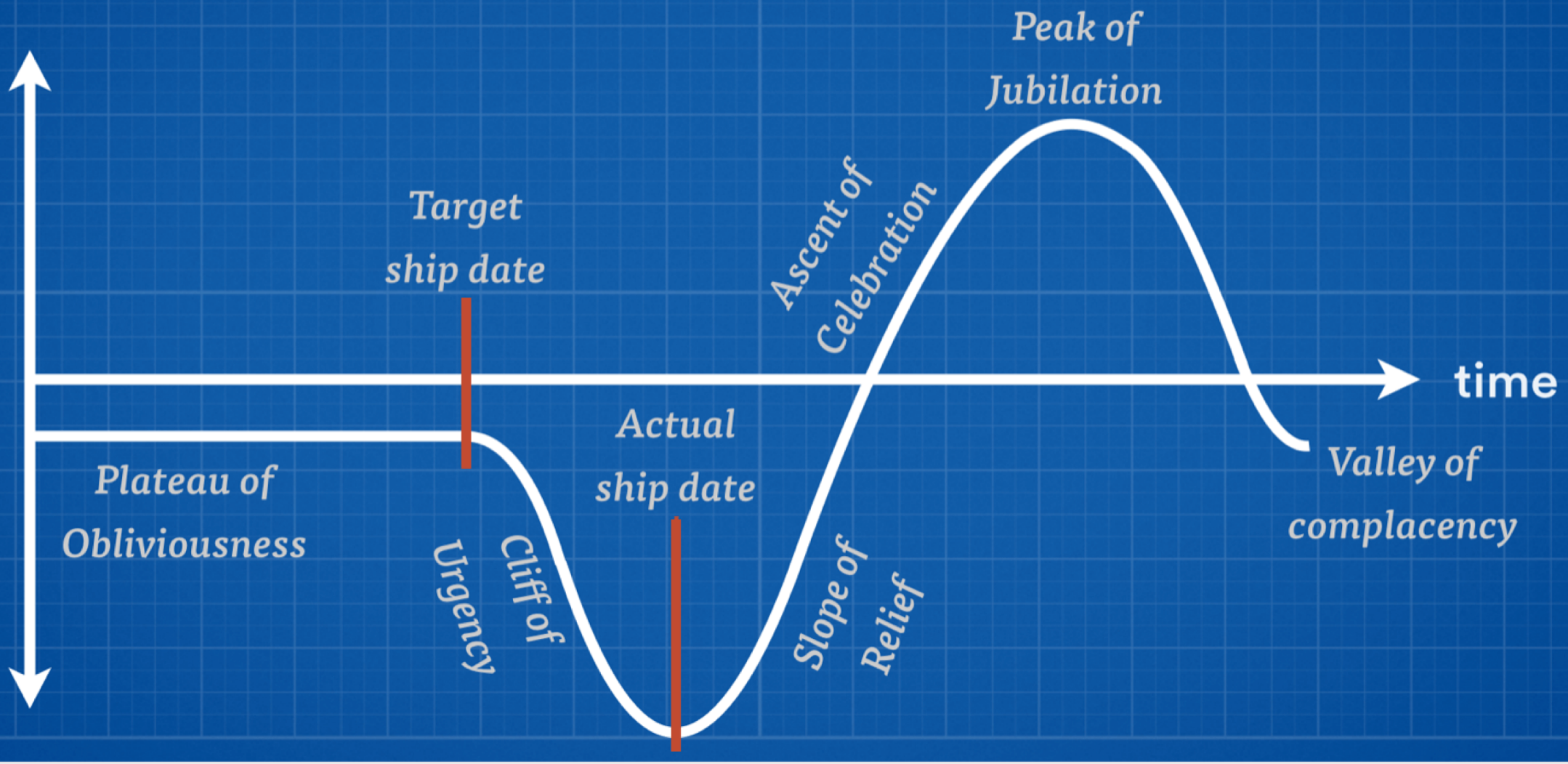
Tejas Parikh (t.parikh@northeastern.edu)
Spring 2018

CSYE 6225
Northeastern University
https://spring2018csye6225.com

Challenges with Traditional Release Method

- Servers must be set up by IT (sometimes manually)
- Third-party software such as application server, etc. must be installed.
- The software artifacts such as EAR or WAR must be copied to the production host.
- Application configuration must be copied or created.
- Finally any reference data needed must be copied over.
- As you can see there are lot of places where things can go wrong.
- With this process, the day of a software release tends to be a tense one.

Emotional cycle of manual delivery



What is Continuous Deployment?

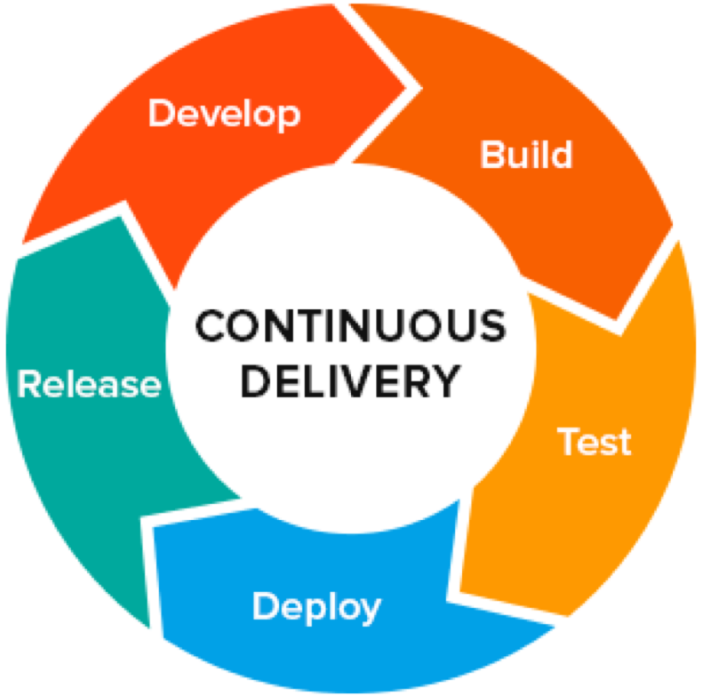
- Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.
- With Continuous Delivery your software is always release-ready, yet the timing of when to push it into production is a business decision, and so the final deployment is a manual step.
- With Continuous Deployment, any updated working version of the application is automatically pushed to production.
- Continuous Deployment mandates Continuous Delivery, but the opposite is not required.

CONTINUOUS DELIVERY

CONTINUOUS DEPLOYMENT

Continuous Delivery

- Continuous delivery is a DevOps software development practice where code changes are automatically built, tested, and prepared for a release to production.
- It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.
- When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.



Continuous Delivery (contd.)

- With continuous delivery, every code change is built, tested, and then pushed to a non-production testing or staging environment.
- There can be multiple, parallel test stages before a production deployment.
- In the last step, the developer approves the update to production when they are ready.

Enabling Continuous Delivery

- Automate
  - The build, deploy, test, and release process bust automated so that it is repeatable.
- Frequent
  - Releases must be frequent.
  - The delta between releases will be small.
  - This significantly reduces the risk associated with releasing and makes it much easier to roll back.

Continuous Deployment

In continuous deployment push to production happens automatically without explicit approval.

AWS CodeDeploy

AWS CodeDeploy

- AWS CodeDeploy is a service that automates code deployments to any instance, including Amazon EC2 instances and instances running on-premises.
- AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications.
- AWS CodeDeploy automates software deployments, eliminating the need for error-prone manual operations enabling you to easily deploy to one instance or thousands.

What AWS CodeDeploy Provides

- Automated Deployments
  - Repeatable Deployments
  - Auto Scaling Integration
- Minimize Downtime
  - Rolling and Blue/Green Updates
  - Stop and Rollback
- Centralized Control
  - Deployment Groups
  - Deployment History
- Easy To Adopt
  - Language and Architecture Agnostic

Continuous Deployment Workflow with GitHub, TravisCI & AWS CodeDeploy

- Commit your changes to GitHub (any branch)
- TravisCI runs your tests
- TravisCI build your artifacts
- TravisCI zips your artifacts and uploads it to AWS S3 bucket
- TravisCI then calls AWS CodeDeploy API to Deploy your Application to your EC2 instance.

Policy for the Server (EC2)

- Policy Name: CodeDeploy-EC2-S3
- This policy will give your EC2 Instance permission to get any data from the s3 service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Policy for TravisCI to Upload to AWS S3

- Policy Name: Travis-Upload-To-S3
- This policy will allow TravisCI to upload your artifacts to AWS S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Policy for TravisCI to Call CodeDeploy

- Name: Travis-Code-Deploy

Create New User for TravisCI

- This user should only have programmatic access and no console access and attach following policies:
- Travis-Upload-To-S3
- Travis-Code-Deploy

Create Role for EC2 Instance

- Name: CodeDeployEC2ServiceRole
- AWS Service Role: Amazon EC2

Create Role for CodeDeploy Application

- Name: CodeDeployServiceRole
- AWS Service Role: Amazon CodeDeploy

Create EC2 Instance

- IAM role must be CodeDeployEC2ServiceRole
- Tag the instance with Key=NAME and Value=CSYE6225

Create S3 Bucket

- Bucket Name: code-deploy.domain-name.tld
  - Example: code-deploy.csye6225.com
- Create the bucket in same region as EC2 instance.

Create Code Deploy Application

- Application name must be same as CODE_DEPLOY_APPLICATION_NAME used when creating IAM policy Travis-Code-Deploy.

Code Deploy Agent Setup (Ubuntu)

- http://docs.aws.amazon.com/codedeploy/latest/userguide/how-to-run-agent-install.html#how-to-run-agent-install-ubuntu

Create AWS CodeDeploy App Spec

- An application specification file (AppSpec file), which is unique to AWS CodeDeploy, is a YAML-formatted file used to:
  - Map the source files in your application revision to their destinations on the instance.
  - Specify custom permissions for deployed files.
  - Specify scripts to be run on each instance at various stages of the deployment process.
- The AppSpec file is used to manage each deployment as a series of lifecycle events. Lifecycle event hooks, which are defined in the file, allow you to run scripts on an instance after most deployment lifecycle events. AWS CodeDeploy runs only those scripts specified in the file, but those scripts can call other scripts on the instance. You can run any type of script as long as it is supported by the operating system running on the instances.http://docs.aws.amazon.com/codedeploy/latest/userguide/writing-app-spec.html

Update TravisCI Configuration File

- Add deployment phase to your TravisCI config file.
- We will deploy to S3 and CodeDeploy

Additional Resources

https://spring2018.csye6225.com/