Continuous Deployment with GitHub, CircleCI, & Amazon CodeDeploy

Tejas Parikh (t.parikh@northeastern.edu)

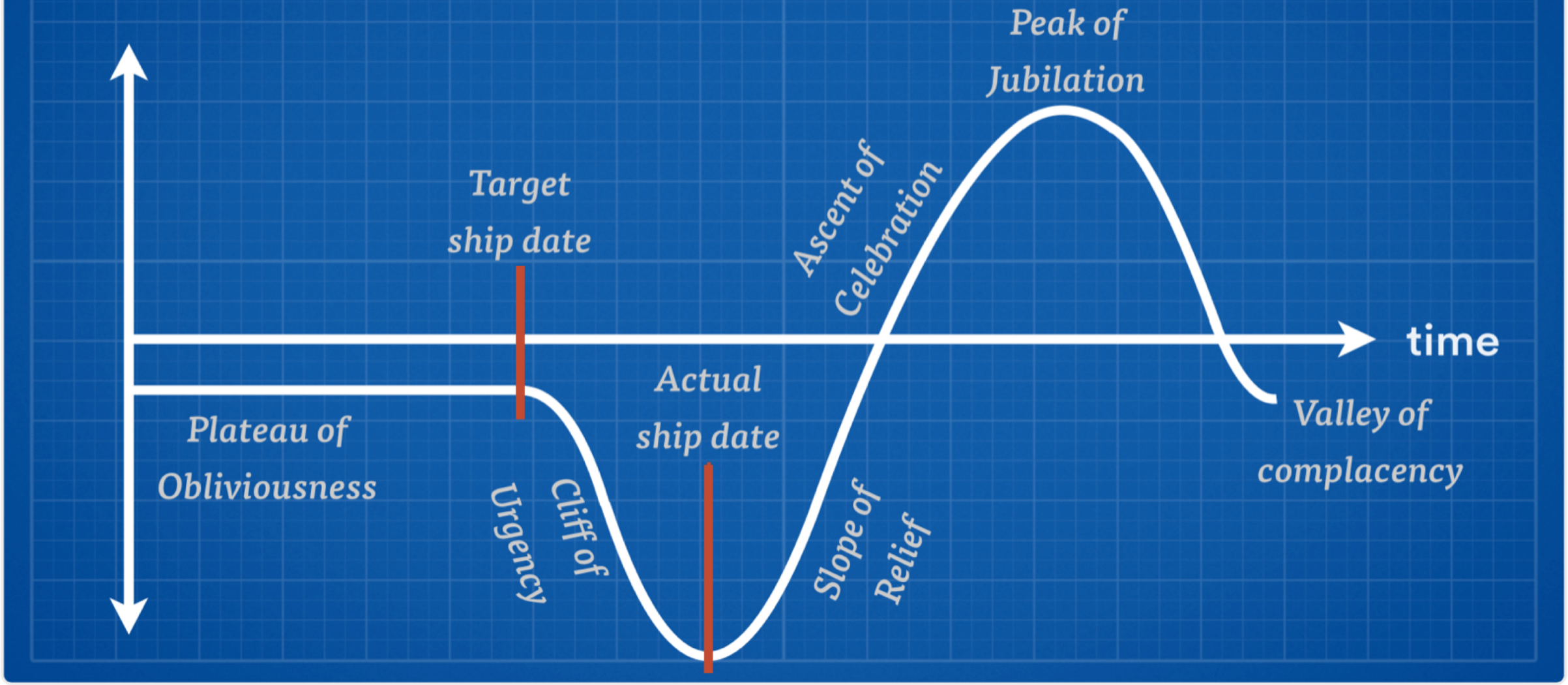
Fall 2019, CSYE 6225

Northeastern University

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?

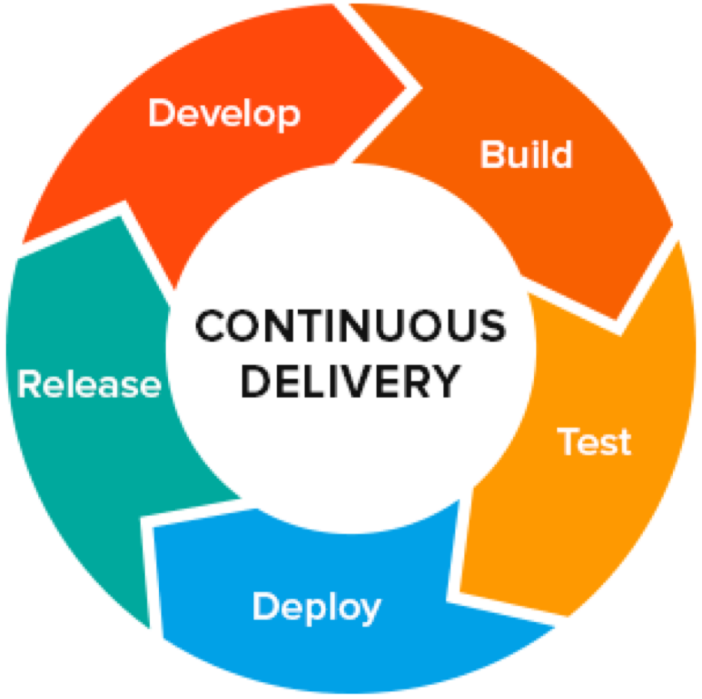
- Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.
- With Continuous Delivery your software is always release-ready, yet the timing of when 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 must be 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.

CircleCI

CircleCI Projects

- A software repository on GitHub is authorized and added as a project to circleci.com.
- Every code change triggers a build and automated tests in a clean container or VM configured for your requirements.

Steps

- Steps are actions that need to be taken to perform your job.
- Steps are usually a collection of executable commands.
- For example, the checkout step checks out the source code for a job over SSH. Then, the run step executes the make test command using a non-login shell by default.

```
    steps:
      - checkout # Special step to checkout your source code
      - run: # Run step to execute commands, see
      # circleci.com/docs/2.0/configuration-reference/#run
          name: Running tests
          command: make test # executable command run in
          # non-login shell with /bin/bash -eo pipefail option
          # by default.
```

Image

- An image is a packaged system that has the instructions for creating a running container.
- The Primary Container is defined by the first image listed in *.circleci/config.yml* file. This is where commands are executed for jobs using the Docker executor.

Jobs

- Jobs are a collection of steps and each job must declare an executor that is either *docker*, *machine*, or *macos*.
- Machine includes a default image if not specified.
- For Docker and macOS you must also declare an image.

Workflows

- Workflows define a list of jobs and their run order.
- It is possible to run jobs in parallel, sequentially, on a schedule, or with a manual gate using an approval job.

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

CodeDeploy Primary Components

Application

- A name that uniquely identifies the application you want to deploy.
- CodeDeploy uses this name, which functions as a container, to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.

Compute Platform

- Compute Platform - The platform on which CodeDeploy deploys an application
  - **EC2/On-Premises** - Describes instances of physical servers that can be Amazon EC2 cloud instances, on-premises servers, or both. Applications created using the EC2/On-Premises compute platform can be composed of executable files, configuration files, images, and more.
  - **AWS Lambda** - Used to deploy applications that consist of an updated version of a Lambda function. AWS Lambda manages the Lambda function in a serverless compute environment made up of a high-availability compute structure. All administration of the compute resources is performed by AWS Lambda. Applications created using the AWS Lambda compute platform can manage the way in which traffic is directed to the updated Lambda function versions during a deployment by choosing a canary, linear, or all-at-once configuration.
  - **Amazon ECS** - Used to deploy an Amazon ECS containerized application as a task set. CodeDeploy performs a blue/green deployment by installing an updated version of the containerized application as a new replacement task set. CodeDeploy reroutes production traffic from the original application, or task set, to the replacement task set. The original task set is terminated after a successful deployment.

Deployment Configuration

- A set of deployment rules and deployment success and failure conditions used by CodeDeploy during a deployment.
- If your deployment uses the EC2/On-Premises compute platform, you can specify the minimum number of healthy instances for the deployment.
- If your deployment uses the AWS Lambda compute platform, you can specify how traffic is routed to your updated Lambda function versions.

Deployment Group

- A set of individual instances.
- A deployment group contains individually tagged instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both.

Deployment Type

- The method used to make the latest application revision available on instances in a deployment group.
- **In-place deployment**: The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated.
- **Blue/green deployment**: The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
  - Instances are provisioned for the replacement environment.
  - The latest application revision is installed on the replacement instances.
  - An optional wait time occurs for activities such as application testing and system verification.
  - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

IAM instance profile

- An IAM role that you attach to your Amazon EC2 instances.
- This profile includes the permissions required to access the Amazon S3 buckets or GitHub repositories where the applications are stored.

Revision

- An EC2/On-Premises deployment revision is an archive file that contains source content (source code, webpages, executable files, and deployment scripts) and an application specification file (AppSpec file).

Service Role

- An IAM role that grants permissions to an AWS service so it can access AWS resources.
- The policies you attach to the service role determine which AWS resources the service can access and the actions it can perform with those resources.

Target Revision

- The most recent version of the application revision that you have uploaded to your repository and want to deploy to the instances in a deployment group.
- In other words, the application revision currently targeted for deployment.
- This is also the revision that is pulled for automatic deployments.

App Spec

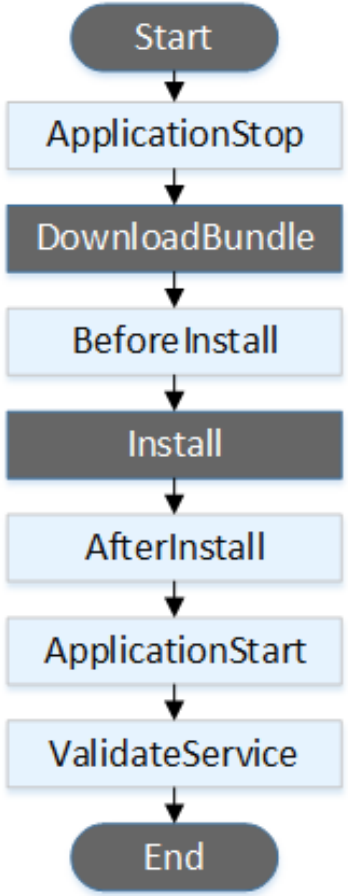
AWS CodeDeploy App Spec

- 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

AppSpec Template

- https://docs.aws.amazon.com/codedeploy/latest/userguide/application-revisions-appspec-file.html#add-appspec-file-server

AppSpec Hooks



AppSpec Hooks - ApplicationStop

- This deployment lifecycle event occurs even before the application revision is downloaded.
- You can specify scripts for this event to gracefully stop the application or remove currently installed packages in preparation of a deployment.
- The AppSpec file and scripts used for this deployment lifecycle event are from the previous successfully deployed application revision.
- **Note** : An AppSpec file does not exist on an instance before you deploy to it. For this reason, the ApplicationStop hook does not run the first time you deploy to the instance. You can use the ApplicationStop hook the second time you deploy to an instance.

DownloadBundle

- During this deployment lifecycle event, the CodeDeploy agent copies the application revision files to a temporary location ***/opt/codedeploy-agent/deployment-root/deployment-group-id/deployment-id/deployment-archive*** folder on Amazon Linux, Ubuntu Server, and RHEL Amazon EC2 instances.

BeforeInstall

- You can use this deployment lifecycle event for preinstall tasks, such as decrypting files and creating a backup of the current version.

Install

- During this deployment lifecycle event, the CodeDeploy agent copies the revision files from the temporary location to the final destination folder. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

AfterInstall

- You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions.

ApplicationStart

- You typically use this deployment lifecycle event to restart services that were stopped during ApplicationStop.

ValidateService

- This is the last deployment lifecycle event.
- It is used to verify the deployment was completed successfully.

Continuous Integration

Building AMIs

Continuous Integration – AWS AMI

- Commits packer template code changes to GitHub repository.
- CircleCI triggers a new build on commit notification.
- CircleCI will run the build steps from CircleCI config file *.circleci/config.yml* from your repository. Build steps does the following:
  - Install *awscli* in your primary container.
  - Download HashiCorp Packer binary and make it executable.
  - Validate packer template.
  - Build AMI and register it with AWS.

Continuous Deployment Workflow with GitHub, CircleCI & AWS CodeDeploy

Build, Test & Deploy Web Application

Continuous Deployment Workflow

- Commits application code changes to GitHub repository.
- CircleCI triggers a new build on commit notification.
- CircleCI will run the build steps from CircleCI config file *.circleci/config.yml* from the repository.
- Build steps should do the following:
  - Install *awscli* in your primary container.
  - Run unit tests.
  - Build artifacts if all tests are successful.
  - Zip your artifacts and upload it to AWS S3 bucket dedicated for code deploy.
  - Call AWS CodeDeploy to deploy the latest revision of your application to the EC2 instances.

Additional Resources

https://fall2019.csye6225.cloud/