DevOps

Best Practices

By Venky

Areas

DevOps can be effective in the following areas:

- CI/CD
- QA
- Release Engineering
- Status
- Monitoring
- Support

CI/CD

- Implement CI/CD Pipeline using Jenkins and GitHub/GitBucket/Blue Ocean
- Pipeline should have several stages for the following:
  - Source file checking for copyright, Lint before building
  - Running unit tests
  - Smoke Tests (A small set of tests to validate minimum functionality)
    - Failure will trigger build failure.
  - Functional tests
    - Complete set of tests to test all features and functionalities
    - Logs of all tests run and provide test report using JUNIT
  - Include a sample set of performance tests
- Build failure should send an email or slack notification to the developer who triggered the build

CI/CD

A successful build should trigger deployment

- Deploy on predetermined platform(s)/Configs (Supported Configs)
- Use CloudFormation(AWS)/Heat(OpenStack)/Deployment Manager(GCP)/ResourceManager(Azure)
- [OR] Terraform(Hashicorp) – cloud agnostic.
  - Deployment as IaC – Parametrize your deployment environment
    - Define Stacks – Staging, Development, Production
    - Write Plan, apply, and destroy
  - Helps repeatability
  - Allows resource monitoring with nomad(in a cluster environment)
  - Failure should send email or slack notification
  - Identified Tests/Certified tests can be run on the deployed config(s)
  - The successful build should send notification to QA for testing

CI/CD and QA

- Any tests automated for a feature, should be integrated in to GitHub for CD
  - This helps catch an issue sooner

SW Configuration and Management:

- Use Ansible/Chef/Salt to automate software provisioning and configuration on the platforms.
- Performance, Stress, Load tests on targeted release builds only

DevOps and Release Engg

- DevOps should define the branches and repos for SW release
- Main Repo [has version info] or Master Branch [a.k.a Production Branch]
  - Maintains the deployment code
  - No high priority issues exist
- Develop Branch [Currently in develop]
- Feature Branches [Each feature has a branch for dev to work]
  - Delete once merged with Develop

DevOps and Dev

- Dev should run lint before merge
  - DevOps to provide SSH/REST/CLI commands or tool(s)
- Add Docker support for building
    - Frontend Java script/AngularJs/NodeJs/JQuery docker
    - Backend Scala or C++ or GoLang or Java docker
    - Vagrant and VirtualBox tools for quick spin up
- Run tests in parallel (if possible to reduce exec time). Shared resource can cause issue.
- Archive test results for major tagged releases (post to Zephyr/TestRail/TM4J or other test management tool)
- Provide Jenkins job(s)
  - For dev to deploy any build as needed
  - Help developers to run tests locally before merge

Status Report

- Builds Status – Jenkins (or CircleCI) Dashboard
- Pipeline Exec Status – BlueOcean Plugin [use Declarative Pipelines]
- QA Test Status Report
  - Pass
  - Fail
  - Performance, Stress/Load tests status for critical builds
  - Generate and send automated test report for builds

Monitoring

- Integrate monitoring tools – Prometheus , Nagios
  - Prometheus
    - Requires exporters to be written for the params to monitor
  - Nagios – Mostly for platform/system resources
  - Grafana
    - Reporting with nice visualization.
    - Identify the queries to design the dashboard(s)/Canvas for monitoring.
- Investigate – ELK (depends on your project/needs)

Support

- DevOps should educate developers/QA on the tools
- Listen and improve those cross functional tools
- DevOps should also work with marketing and Sales for customer enablement

Finally…

It is a about *culture, discipline and practices* across the development, QA and Operations that make the product/service and business be successful.

Questions