he following are some best practice dos and don’ts for CI/CD. Do: Amazon Web Services – Practicing CI/CD on AWS Page 28 • Treat your infrastructure as code o Use version control for your infrastructure code. o Make use of bug tracking/ticketing systems. o Have peers review changes before applying them. o Establish infrastructure code patterns/designs. o Test infrastructure changes like code changes. • Put developers into integrated teams of no more than 12 self-sustaining members. • Have all developers commit code to the main trunk frequently, with no long-running feature branches. • Consistently adopt a build system such as Maven or Gradle across your organization and standardize builds. • Have developers build unit tests toward 100% coverage of the code base. • Ensure that unit tests are 70% of the overall testing in duration, number, and scope. • Ensure that unit tests are up-to-date and not neglected. Unit test failures should be fixed, not bypassed. • Treat your continuous delivery configuration as code. • Establish role-based security controls (that is, who can do what and when). o Monitor/track every resource possible. o Alert on services, availability, and response times. o Capture, learn, and improve. o Share access with everyone on the team. o Plan metrics and monitoring into the lifecycle. • Keep and track standard metrics. o Number of builds. o Number of deployments. Amazon Web Services – Practicing CI/CD on AWS Page 29 o Average time for changes to reach production. o Average time from first pipeline stage to each stage. o Number of changes reaching production. o Average build time. • Use multiple distinct pipelines for each branch and team. Don’t: • Have long-running branches with large complicated merges. • Have manual tests. • Have manual approval processes, gates, code reviews, and security reviews.